Ph.D. DISSERTATION

DESIGN METHODOLOGY FOR MESH BASED CLOCK NETWORKS

메쉬 기반의 클락 네트워크 설계 방법론

BY

Minseok Kang

FEBRUARY 2015

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

DESIGN METHODOLOGY FOR MESH BASED CLOCK NETWORKS

메쉬 기반의 클락 네트워크 설계 방법론

BY

Minseok Kang

FEBRUARY 2015

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

DESIGN METHODOLOGY FOR MESH BASED CLOCK NETWORKS

메쉬 기반의 클락 네트워크 설계 방법론

지도교수 김 태 환

이 논문을 공학박사 학위논문으로 제출함

2014년 11월

서울대학교 대학원

전기컴퓨터 공학부

강 민 석

강민석의 공학박사 학위 논문을 인준함

2014년 12월

위 원 장: _____

부위원장: _____

위　　원: _____

위　　원: _____

위　　원: _____

Abstract

The clock distribution network in a synchronous digital circuit delivers a clock signal to every storage element i.e., clock sink in the circuit. However, since the continued technology scaling increases PVT (process-voltage-temperature) variation, the increase of clock skew variation is highly likely to cause performance degradation or system failure at run time. Recently, to mitigate the clock skew variation, many researchers have taken a profound interest in the clock mesh network. However, though the structure of clock mesh network is excellent in tolerating timing variation, it demands significantly high power consumption due to the use of excessive mesh wire and buffer resources. Thus, optimizing the resources required in the mesh clock synthesis while maintaining the variation tolerance is crucially important. The three major tasks that greatly affect the cost of resulting clock mesh are (1) *mesh segment allocation*, (2) *mesh buffer allocation and sizing*, and (3) *clock sink binding to mesh segments*. Previous clock mesh optimization approaches solve the three tasks sequentially, one by one at a time, to manage the run time complexity of the tasks at the expense of losing the quality of results. However, since the three tasks are tightly inter-related, simultaneously optimizing all three tasks is essential, if the run time is ever permitted, to synthesize an economical clock mesh network. In this dissertation, we propose an approach which is able to tackle the problem in an integrated fashion by combining the three tasks into an iterative framework of incremental updates and *solving them simultaneously* to find a globally optimal allocation of mesh resources while taking into account the clock skew tolerance constraints. The core parts of this dissertation are a precise analysis on the relation among the resource optimization tasks and an establishment of mechanism for effective and efficient integration of the tasks. In particular, to handle the run time problem, we propose a set of speed-up techniques i.e., *modeling*

RC circuit for eliminating redundant matrix multiplications, *exploiting sliding window scheme*, and *fast buffer sizing effect estimation*, which are fitted into our context of fast clock skew estimation in mesh resource optimization as well as an invention of early decision policies. In summary, this dissertation presents the efficient design methodology for clock mesh synthesis with consideration on integration of three tasks and reduction of runtime complexity.

keywords: VLSI&CAD, clock mesh synthesis, variation tolerance, resource allocation & binding

student number: 2010-30206

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

With the aggressive scaling down of CMOS process technology, the performance and density of integrated circuit (IC) chips had continuously increased in the last decades. However in recent years, the IC industry has been facing the challenges which inhibit the sustainable progress of CMOS technology [7]. The manufacturing and power-thermal challenges are the most urgent issues in electronic design automation (EDA).

In manufacturing process, there are several variation sources affecting design parameters of circuit [8–10]. The gate length, gate width and gate oxide thickness are sources of device variation and metal thickness, inter-layer dielectric thickness and linewidth are sources of interconnect variation. The process variation in design parameters directly affects the performance and yield of IC chips. The process variation will present the significant impact on critical-path delay of manufactured chips, and hence incurs timing uncertainty in estimating the minimum clock period, T_{min} in Eq. 1.1.

$$T_{min} = t_{cq} + D_{crit} + t_{st} + T_{skew} \tag{1.1}$$

where t_{cq} is propagation delay of F/F (clock to Q delay), D_{crit} is critical-path delay, t_{st} is setup time of the F/F and T_{skew} is clock skew of the clock distribution network. Therefore, the designers who work on timing signoff add the timing margin to compensate the uncertainty caused by process variation by sacrificing performance. Because insufficient timing margin causes reduction in the chip yield, the impact of process variation should be considered at every stages of circuit design to minimize the timing margin while maintaining the chip yield. The clock skew should be also minimized to improve the performance of the circuit. Furthermore, without considering the effect of process variation on clock skew, timing violation of the manufactured chips increase which may lead to lowered yield of chips.

The increasing of power consumption and power density and the lack of advanced battery and cooling system are the major challenges that hinder the continuous growth of chip performance [11–14]. Dynamic power consumption of a synchronous digital circuit is defined to as follows:

$$P = \alpha \cdot C \cdot V^2 \cdot f \tag{1.2}$$

where α is switching activity of the circuit, C is total capacitance of circuit, V is operating voltage and f is the clock frequency. Many techniques have been suggested in different levels of design abstraction to minimize the each factor in Eq. 1.2. Power gating, clock gating and dynamic voltage / frequency scaling (DVFS) techniques minimize the power consumption of circuit in idle state, while other optimization techniques achieve the power reduction by minimizing the capacitance of circuit. As the clock frequency and size of the clock network increase, the portion of power consumption of the clock distribution network has increased substantially. It was reported that the clock distribution network consumes up to 40% of the total power in high speed processors [15, 16]. Consequently, the power consumption of circuits can be reduced greatly by optimizing the clock distribution network.

Clock meshes are one of promising solution to minimize the effect of variations on the clock skew due to the redundancy of the grid structure. However, the high power consumption of the massive parasitic capacitance of mesh wires hinders their wide adoption in the industry. In this dissertation, we propose a clock mesh optimization framework while considering the variation tolerance constraints to maintain the variation tolerance of the clock mesh.

1.2 Contributions of This Dissertation

To overcome the limitations of the conventional clock mesh optimizations, we propose a comprehensive clock mesh optimization method that globally optimizes three resource optimization problems, such as *mesh segment allocation*, *mesh buffer allocation and sizing* and *stub binding* while satisfying the variation tolerance constraints in this dissertation. Due to the time complexity of verifying the variation tolerance of every resource perturbation, we adopt a set of computation techniques and exploit it to our resource optimization framework. The contributions of this dissertation are summarized as follows:

1. We integrate three resource optimization problems into a resource cost of the clock mesh and present the integrated clock mesh optimization algorithm that iteratively applies the minimum cost movement while meeting the variation tolerance constraints. By solving the three problems simultaneously, the resource of the clock mesh can be globally minimized in comparison with previous works that sequentially solves the three problems.

2. We devise the variation tolerance constraints, such as clock skew and multiple path constraints, to reflect the effect of a movement on variation tolerance of the clock mesh. We suggest a new formulation of multiple path constraint based

on *the degree of 'collective' path redundancy* to overcome the weakness of the conventional multiple path constraint.

3. To deal with the runtime complexity of calculating the effect of every resource perturbation on the clock skew, we adopt and fully exploit a set of efficient computation techniques: *modeling RC circuit for eliminating redundant matrix multiplications*, *exploiting sliding window scheme*, and *fast buffer sizing effect estimation*.

Chapter 2

Background

2.1 Clock Distribution Network

For synchronous digital circuits, clock distribution network is constructed to transfer the clock signal to sequential elements such as flip-flops (FFs) or latches. Fig. 2.1 depicts a digital circuit synchronized by a clock network. The red lines are clock distribution network from clock source to each FF. The clock buffers are inserted on the paths due to the long path from clock source to FF. A clock signal is generated by a clock generator, phase locked loop (PLL), and fed to a single pin input of the chip which is called *clock source* in clock network synthesis. The clock signal oscillates between ground and supply voltage in each cycle. Every FF is triggered on each rising (falling) edge of clock signal and captures the input value. After capturing the input value, each FF holds the output value until the next rising (falling) edge of clock signal. The FFs are called *clock sinks* in clock network synthesis.

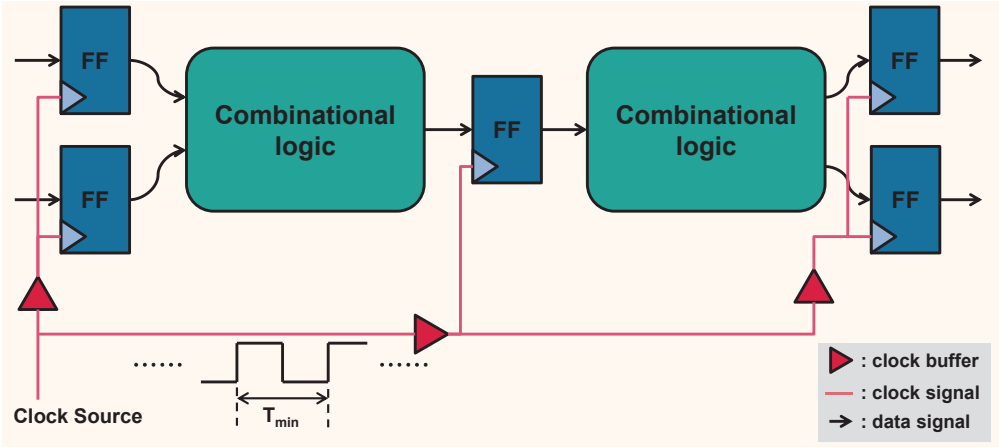


Figure 2.1: A synchronous digital circuit.

2.2 Clock Network Topologies

The clock network topologies can be briefly sorted into four types: tree, spine, mesh and hybrid. Fig. 2.2 shows a simple example of tree, spine and mesh topologies. Fig. 2.2 (a) is a symmetric H-tree topology. Because each path from the clock source to the clock sinks has identical setup of wires and buffers, the signal delay to each clock sinks is also identical in nominal condition. For the same reason, H-tree provides relatively good variation tolerance than clock tree topology. however, the limitation of H-trees is that it is difficult to create and optimize an H-tree for irregularly distributed clock sinks. Fig. 2.2 (b) is a binary clock tree topology. Due to their convenience of analysis and optimization, the clock tree topology is commonly used in automatic synthesis. It has the advantage in resource optimization but is highly sensitive to variations in wires and buffers.

In clock spine topology, multiple spines are placed across the chip and clock sinks are connected to the closest spine. Although there will be residual skew due to the asymmetry of local connections to the clock sinks, the balanced clock signal in a spine

provides variation tolerance.

Fig. 2.2 (d) is a clock mesh topology. A clock mesh is composed of a rectangular cells of wire grid called clock mesh, buffers that directly drive the clock mesh, and a top level clock tree that drives the buffers. The clock sinks are connected to the mesh segments which is nearest to the sinks in the clock mesh. The clock mesh shows low clock skew and high variation tolerance, but the power consumption of the grid wires is a major disadvantage of the clock mesh.

The hybrid clock topologies are used to overcome the limitations of primary topologies described above by combining the primary topologies. The clock tree with cross link insertion and clock mesh with local clock tree are the commonly studied hybrid clock topologies.

2.3 Design Metrics of Clock Network

Clock frequency: Clock frequency represents the number of oscillations of clock signal in a second. Because speed of the data transfer in synchronous digital circuits depends on the clock frequency, the clock frequency is often used as the performance metric of the digital circuits.

Power: The power consumption of a clock network is proportional to the load capacitance and clock frequency of the clock network. Most conventional approaches of clock power minimization have concentrated on reducing capacitive load and switching activity of the clock networks. Examples are the clock tree resource optimization (e.g., [1, 17, 18]), clock mesh resource optimization (e.g., [3, 4, 19]), buffer insertion(e.g., [20, 21]) and clock gating (e.g., [22, 23]). The resonant clock network reduces the clock power consumption in a different way from conventional approaches [24–26]. In resonant clock network, the electrons are stored in the embedded inductors instead of dissipating through nMOS elements in clock buffers.

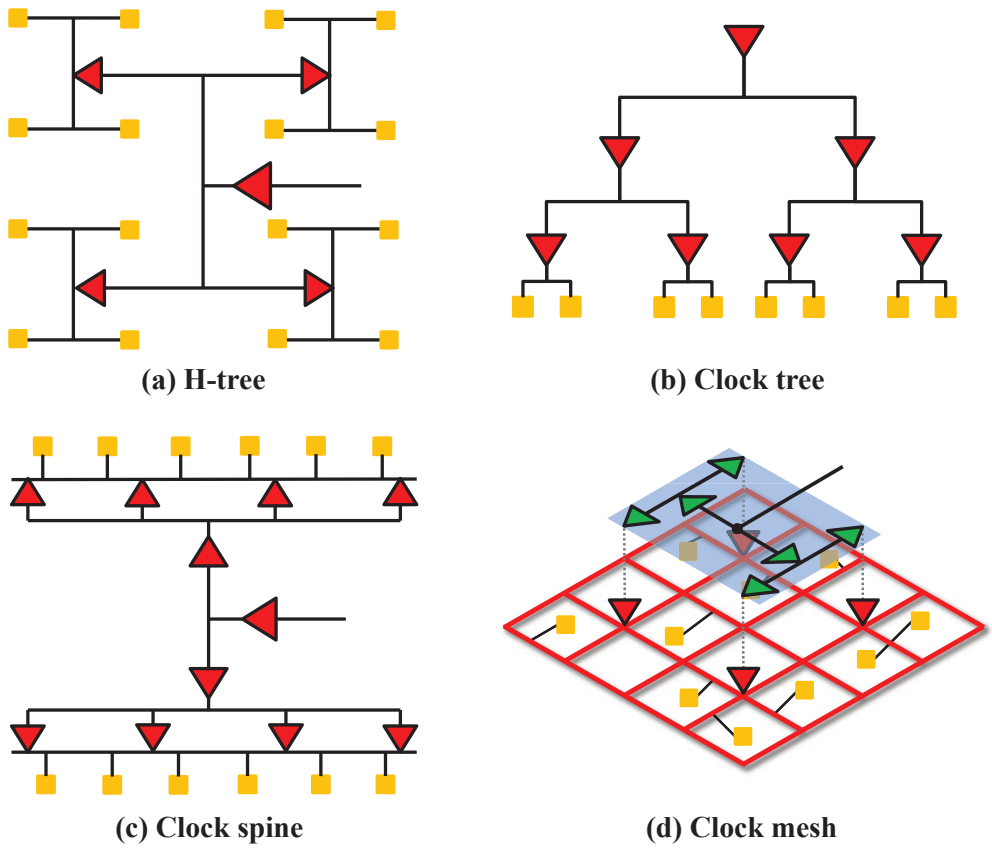


Figure 2.2: Clock network topologies: (a) H-tree, (b) general clock tree, (c) clock spine and (d) clock mesh.

Clock skew: The clock skew in a clock network refers to the difference of the maximum and minimum arrival times to clock sinks in the clock network. Since a large clock skew causes circuit performance degradation or a system failure, the most important goal in the clock network design and synthesis is to meet the clock skew constraint. For this reason, many studies have focused on designing or optimizing the clock network under various clock skew constraints such as zero skew (e.g., [1, 27–29]), bounded skew (e.g., [17, 30]), and useful skew (e.g., [18, 31]).

Slew rate: The slew rate of the clock signal is defined as the transition time from V_l to V_h. Normally, the V_l and V_h are 10% and 90% of supply voltage, respectively. The high slew rate of clock signal causes the increase of delay and power consumption of downstream gates. Thus, the slew rate should be constrained in clock network synthesis [3, 32].

2.4 The Effects of Variations on Clock Skew

For the manufacturing side, since the process variation is rapidly increasing as the technology node shrinks, it becomes unacceptable to treat the clock skew as a deterministic value. More precisely, the continued technology scaling to sub-100nm regime has made it more difficult to predict the clock skew of the individual fabricated chips. For example, the work in [8, 33–37] showed that the transistor parameters such as channel length/width, oxide thickness, threshold voltage, and wire width/thickness across the chip significantly vary. According to [33], the delay variation of interconnect can cause up to 25% variation of the clock skew. It is also reported in [8] that the clock skew of H-tree can grow up to about 30% of the clock period in 50nm regime. Moreover, during the circuit execution, the clock skew variation is aggravated because of the environmental variation such as the fluctuations of temperature and supply voltage.

We analyzed nominal and worst case clock skew of clock tree for ISCAS89 bench-

mark circuits [2] in Fig. 2.3. The clock trees are produced by algorithm in [1]. We used the same technology and variation parameters used in Section 4.5. The worst case clock skew is 101.1ps on average which is up to 6X larger than the nominal clock skew. Although the worst case clock skew depends on the assumed technology and variation parameters, the results in Fig. 2.3 shows that the clock skew variation should be a major concern in clock network synthesis.

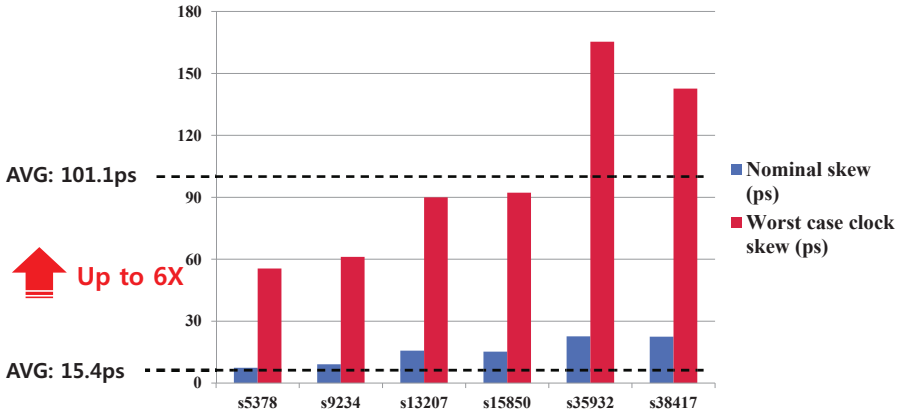


Figure 2.3: The worst case clock skew analysis of clock trees implemented by [1] on ISCAS89 benchmark circuits [2].

Various works have been proposed for clock network structures in order to mitigate or tolerate the clock skew variation. The structures are roughly classified into *variation aware clock tree structure* (e.g., [38, 39]), *non-tree structure by inserting links* (e.g., [40–42]) and *clock mesh network* (e.g., [3, 4, 19, 43–46]). Among the structures, the clock mesh network offers the highest tolerance to the delay variation since the mesh grid contains multiple paths from the clock source to every clock sink, thus, the delay variation in a path is compensated by the delay of another path(s). Though the clock mesh network has a strong advantage of high delay variation tolerance, there is one critical limitation that hinders the clock mesh from a wide adoption in the industry,

which is the high power dissipation due to the massive parasitic capacitance of the mesh wires.

To demonstrate the advantage and disadvantage of the clock mesh in comparison with clock tree, which is the most popular clock network due to the convenience in timing analysis and optimization, we analyzed the worst case clock skew and power consumption of the results by conventional clock tree synthesis algorithm [1] and clock mesh synthesis algorithm [3] for ISCAS89 benchmark circuits [2]. We measure the worst case clock skew and power consumption by Hspice Monte-Carlo simulation and Hspice simulation, respectively. We use the same technology and variation parameters used in Section 4.5. Fig. 2.4 shows the ratio of worst case clock skew (blue) and power consumption (red) between results of clock mesh and clock tree. The clock mesh shows the 44.8% lower worst case clock skew while consuming 25.5% more power than the clock tree. It is clear that the advantage of the clock mesh is high variation tolerance and the disadvantage is high resource usage. We introduce conventional clock mesh synthesis flow in next section, especially the clock mesh optimization which minimizes the resource usage of the clock mesh while maintaining the variation tolerance.

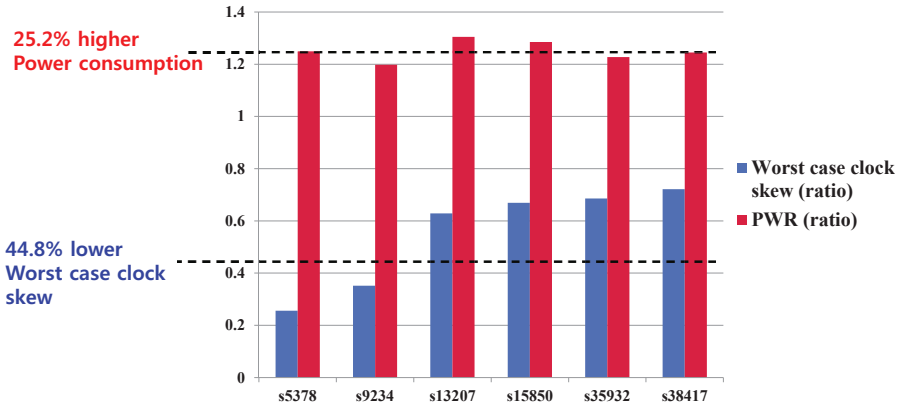


Figure 2.4: The comparison of worst case clock skew and power consumption between clock tree [1] and clock mesh [3] implemented for ISCAS89 benchmark circuits [2]

Chapter 3

Clock Mesh Synthesis Flow

3.1 Elements of Clock Mesh

As depicted in Fig. 3.1, clock mesh consists of *mesh segments*, driving *mesh buffers*, and clock sinks connected to *mesh segments* by *stub wires*. Each element is closely related in variation tolerance of the clock mesh. The resource of clock mesh can be globally optimized by considering the influence of each element together.

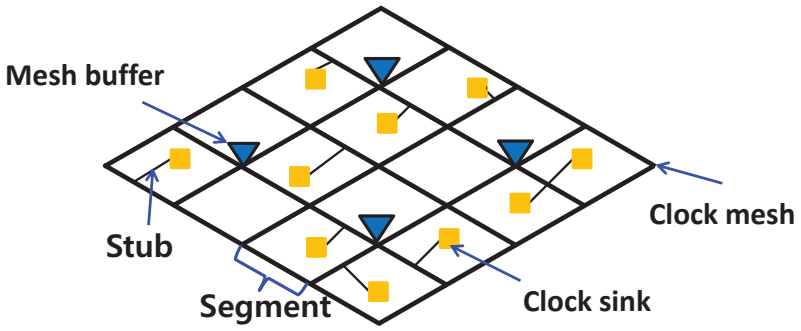


Figure 3.1: Structure of clock mesh.

3.2 Conventional Clock Mesh Synthesis Overview

The clock mesh synthesis includes the mesh grid generation, clock sink binding, buffer placement and sizing, and clock mesh optimization. Fig. 3.2 shows the conventional clock mesh synthesis flow. Each step of clock mesh synthesis is performed sequentially in previous works in general. First, given a mesh size and sink locations, a mesh grid is generated and clock sinks are directly connected to the mesh segments by stub wires. After the mesh grid generation, mesh buffers placement and sizing are performed simultaneously to minimize mesh buffer area while the clock slew constraint is met. The clock mesh optimization is composed of three tasks (i.e., clock mesh reduction, clock sink rebinding and mesh buffer resizing). The unnecessary mesh segments are removed in clock mesh reduction. The clock sinks bound to removed mesh segments are rebound to nearby mesh segments. The size of mesh buffers are adjusted to compensate the capacitance of removed mesh segments and rebound stub wires. An example illustrating the steps of clock mesh synthesis is depicted in Fig. 3.3.

3.3 Initial Grid Generation

The initial clock mesh generation consists of generating the mesh grid with size of $n \times m$ and clock sink binding where n and m is number of horizontal and vertical nodes of the clock mesh. [4] suggested the mesh size decision algorithm (called mesh planning) under clock skew constraint while many works have assumed the mesh size as user input. Due to the lack of information that will be decided in the next steps such as mesh buffer placement and sizing and clock mesh optimization, they calculate the worst clock skew by assuming the worst case placement of mesh buffers and stub binding. [44, 46] proposed the ILP based mesh line allocation algorithm in which they decide mesh lines to allocate under timing constraint using delay approximate model suggested in each papers. [19] proposed an algorithm of synthesizing a non-uniform

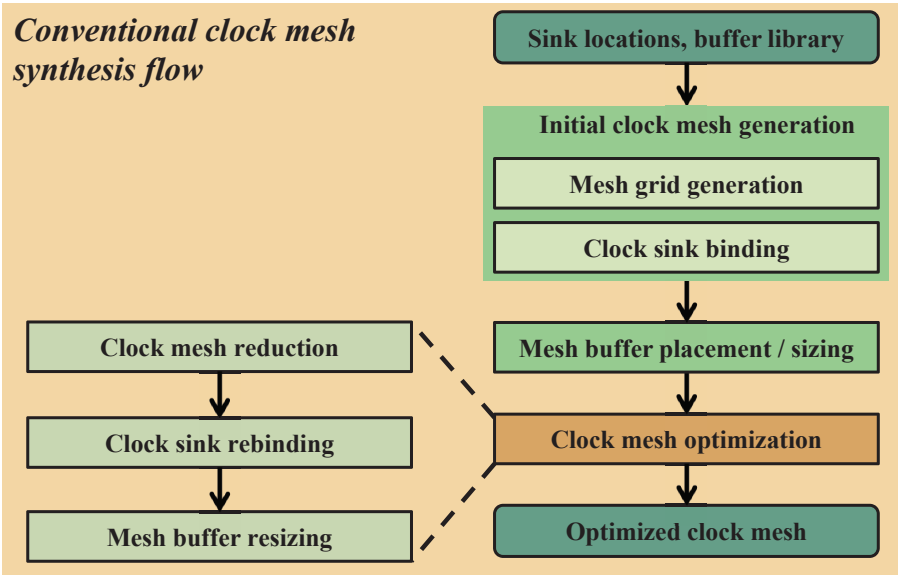


Figure 3.2: The conventional clock mesh synthesis flow

clock mesh by shifting horizontal and vertical mesh lines to reduce the wirelength of stub wires.

Sink binding problem has not been widely addressed in conventional clock mesh synthesis approaches. While most of the works assume that the clock sinks are connected to closest mesh segments, [44] suggested ILP based clock sink binding algorithm to balance the load capacitance of each mesh segments.

3.4 Mesh Buffer Placement and Sizing

The mesh buffer placement and sizing algorithm was firstly suggested in [3]. They suggested the set-cover based mesh buffer placement and sizing algorithm under slew constraint. Given buffer library and clock mesh, the covering region (CR) of a buffer placed on a mesh node of the clock mesh is defined as the set mesh nodes that total sum of capacitance of the nodes is not exceed the maximum load capacitance of the

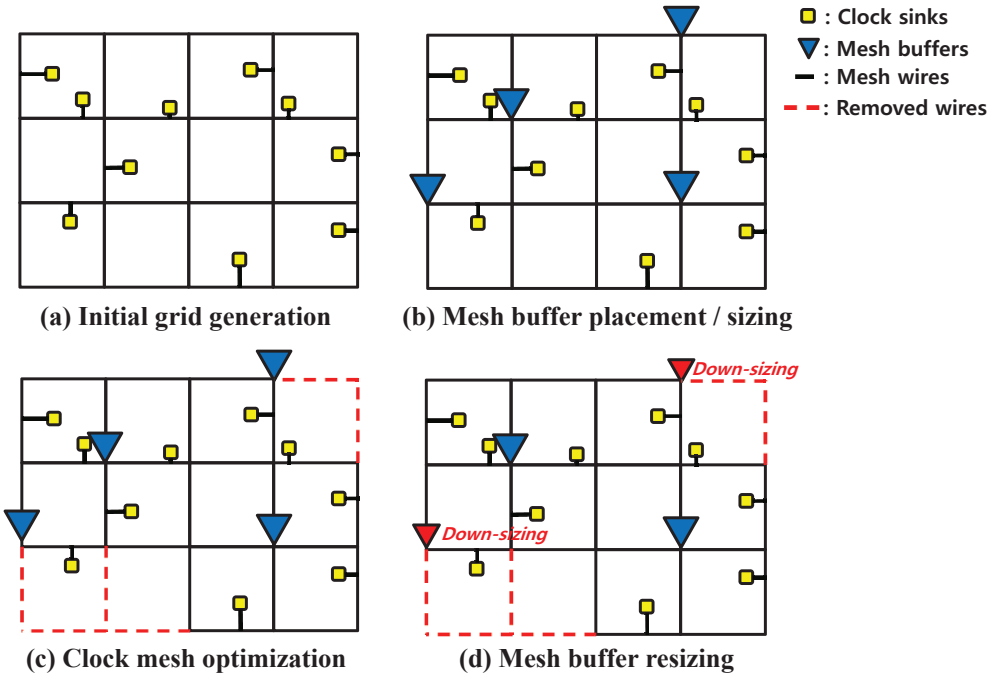
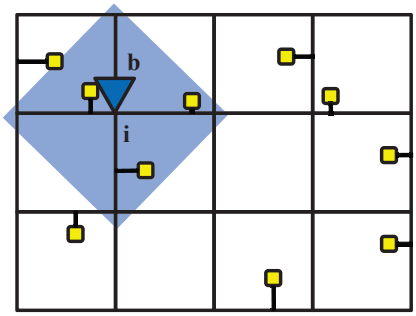


Figure 3.3: An example illustrating the steps of clock mesh synthesis

buffer. [3] presented the greedy method which iteratively places a minimum cost buffer to minimize the total buffering area until every node of clock mesh is covered by more than one buffer.



(a) An example of covering region

$I = (i_1, i_2, \ldots, i_{n\times m})$	*Set of nodes; given mesh size* $(n \times m)$
$BL=(b_1, b_2, \ldots, b_p)$	*Set of buffers in given buffer library;* $N(BL) = p$
$CAP(i)$	*Total sum of wire and sink capacitance around node i, where* $i \in I$
$CAP(b)$	*Maximum capacitance load of a buffer b under output slew constraint (i.e., 100ps), where* $b \in B$
CR_i^b	*Covering region (subset of I) of a buffer* $b \in B$ *placed at node* $i \in I$, *where* $\Sigma_{j\in CR}\, CAP(j) \leq CAP(b)$

(b) Notations of covering region

Figure 3.4: An example and notations of covering region when buffer type *b* of given buffer library is placed at mesh node *i* of given clock mesh

[4] proposed the improved cost model for mesh buffer placement and sizing algorithm suggested in [3] considering RC signal attenuation. [19] proposed LP based mesh buffer sizing algorithm which resizes mesh buffers until the clock skew constraint is met. The proposed algorithm shows good quality solution with the disadvantage of high time consumption.

3.5 Clock Mesh Optimization

Clock mesh optimization consists of clock mesh reduction, clock sink rebinding and mesh buffer resizing. To form a grid structure, the clock mesh potentially has unnecessary mesh segments which can be deallocated with small impact on variation tolerance of the clock mesh. These mesh segments can be deallocated by clock mesh reduction. The network survivability based mesh reduction and network sensitivity based mesh reduction algorithm was proposed in [3] and [4], respectively.

The load capacitance driven by each mesh buffer is reduced after clock mesh reduction. The unbalanced load capacitance of mesh buffers will increase the clock skew. Therefore, the mesh buffers placed nearby the dellocated mesh segments should be resized to reduce the clock skew and resource usage. [3, 4] suggested a greedy buffer resizing algorithm which iteratively downsizes the mesh buffers while maintaining the covering range of mesh buffers.

We summarize the optimization criteria, problems which isn't considered in each algorithm and optimization constraint of the two conventional clock mesh optimization flows i.e., MeshWorks [4] and NS [3] in Fig. 3.5. We propose a clock mesh optimization flow which overcomes the limitations of previous works (highlighted words in red in Fig. 3.5) in next chapter.

	MeshWorks	NS
Optimization criteria	**Network sensitivity (clock skew)** $\mathbf{Cost(Seg_i) = Max(\frac{\partial Del_j}{\partial W(Seg_i)} - \frac{\partial Del_k}{\partial W(Seg_i)})W(Seg_i)}$ $\mathbf{, \forall\, j, k \in sinks}$ ***Maximum clock skew fluctuation*** **by decreasing width of Seg_i.**	**Network survivability (p, q, L_{bound})** **Maintaining *q edge disjoint path* ($l < L_{bound}$) from *each p mesh buffers* for every clock sinks**
Clock skew	**Considered**	***Not considered***
Path redundancy	***Not considered***	**considered**
Stub rebinding	***Not considered***	***Not considered***
Constraint	**Wirelength constraint**	**(p, q, L_{bound})**
Approach	***Sequential approach*** **(buffering → optimization → resizing)**	

Figure 3.5: The summary of two conventional clock mesh optimization algorithms i.e., MeshWorks [4] and NS [3].

Chapter 4

Integrated Resource Allocation and Binding in Clock Mesh Synthesis

4.1 Introduction

In the last few years, a number of works have addressed the problem of optimizing clock mesh resources to be allocated, so that the power consumption can be minimized [3, 4, 19, 44, 46]. Cho, Pan, and Puri [44] proposed a clock mesh construction method that is driven by the horizontal and vertical mesh line allocation (equivalently, called mesh line selection) under clock skew constraint where they used an RC network model to estimate the clock skew of every instance of clock mesh explored. They employed a 0-1 integer linear programming (ILP) to formulate the mesh line allocation. In addition, they considered the assignment of clock sinks to mesh segments in a way to balance the load capacitance of mesh segments. However, the work has a few important items that have not been fully addressed: it performed buffer sizing iteratively by repeatedly running (slow) Spice simulation and spent too much time to solve the 0-1 integer linear programming even though they used a clock skew estimation model. On

the other hand, Guthaus, *et al.* [19] proposed a method of synthesizing a non-uniform clock mesh by shifting horizontal and vertical mesh lines to reduce the total length of *stub* wires, which refer to the wires connecting clock sinks to mesh segments. They employed a linear programming (LP) for sizing buffers, in which they applied the linear programming repeatedly to resize mesh buffers until the clock skew constraint is met. The method produced a good quality of buffer sizing at the expense of run time. Here, they did not attempt to optimize the mesh structure. Lu, Mao, and Taskin [46] proposed an incremental register placement and timing slack aware clock mesh generation algorithm. They evaluated the feasible moving region (FMR) of each register by the timing slack of each register. Registers can move without timing violation if they are within their FMRs. They generated the whole clock mesh to minimize total wirelength considering FMR of each register. The clock mesh optimization (or reduction) problem was first addressed by Venkataraman, Feng, and Li in [3]. They suggested a survivable network theory based clock mesh reduction and a simultaneous mesh buffer placement and sizing based on a greedy set cover. However, meeting the clock skew constraint was not taken into account through the work. Moreover, mesh buffering and mesh reduction were carried out sequentially. On the other hand, Rajaram and Pan [4] performed an initial mesh planning, in which they expressed the relation between total wirelength and clock skew roughly as a function of clock mesh size, by which they determined the size of clock mesh with a minimum total wirelength under the clock skew constraint. In addition, they introduced a network sensitivity based clock mesh reduction, in which the delay sensitivity of each sink is calculated in terms of the width of mesh segments. Based on the computed delay sensitivity, they removed the mesh segments with less effect on clock skew. Rather than computing an estimation of the actual resulting clock skew during the process of incremental clock mesh reduction, they used the resource bound as stopping criteria of the clock mesh reduction. Consequently, it is not sure that the resulting clock skew after the execution of clock mesh

reduction always satisfies the clock skew constraint. Another drawback is that since removing one mesh segment affects the delay sensitivity of the neighbor mesh segments, the delay sensitivity recalculation will be required for every removal of mesh segment. Furthermore, like the work in [3], this work inherently follows a sequential optimization flow.

This work overcomes the key limitations of the aforementioned previous works [3, 4, 19, 44]. More precisely, the objective of our proposed approach of comprehensive allocation and binding of clock mesh resources is to minimize the clock mesh resources by solving the three major mesh optimization tasks simultaneously by inventing a set of computation acceleration techniques to strictly take into account the satisfaction of clock skew tolerance constraints, while providing the additional advantage of flexibility to be fitted into diverse iterative optimization flows. The new features of our work can be summarized as:

- We place our *primary concern on meeting the clock skew tolerance constraints* in the process of clock mesh synthesis. We devise, in our clock mesh resource optimization, a number of efficient computation techniques: *modeling RC circuit for eliminating redundant matrix multiplications*, *exploiting sliding window scheme*, and *fast buffer sizing effect estimation*. We exploit those delay computation techniques to accurately measure the effect of every resource perturbation on the clock skew during our mesh resource optimization framework.

- We solve the three resource optimization problems namely *mesh segment allocation*, *mesh buffer allocation and sizing*, and *clock sink (or stub) binding to mesh segments* simultaneously, so that the clock mesh resources should be minimized globally while meeting the clock skew tolerance constraints. Since the three problems are tightly inter-related, simultaneously solving the three problems is essential to produce a mesh structure of minimal cost.

- We devise a new formulation of the variation tolerance constraint based on so called *the degree of 'collective' path redundancy* to overcome the weakness of the conventional multiple path constraint, in which the number of disjoint paths to a sink from *each* driving buffer in a certain distance should strictly exceed a given threshold even though there may exist nearby buffer(s) that can compensate for the less variation tolerance by the paths.

- We develop a framework of clock mesh synthesizer which iteratively refines the three clock mesh optimizations simultaneously while satisfying the clock skew tolerance constraints. We design our proposed framework to be easily fitted into any iterative optimization flow only if at each iteration an incremental change of resource allocation and binding is tried.

- We implemented the existing state-of-the-art mesh synthesis algorithms and compared their performance with ours in terms of various measurements such as resource usage, clock skew variation, power consumption, and run time. This work also provides a useful knowledge about what factors the existing and our mesh synthesis works emphasize and how they approach.

The rest of the work is organized as follows. Section 4.2 introduces a set of important observations regarding the relation between the mesh resource allocation and clock skew, of which the existing works have never or not been fully aware. Then, motivated by the observations, we develop an iterative resource allocation and binding framework for the synthesis of clock mesh networks in section 4.3. Subsequently, section 4.4 details the computation acceleration techniques we have devised for the fast clock skew estimation that is essential to any (iterative) clock mesh synthesis framework. Section 4.5 includes the experimental setup for generating initial clock mesh networks, followed by diverse experimental results to assess how much our proposed approach is effective in optimizing the clock mesh resources while preserving the clock

skew tolerance constraints. Finally, concluding remarks of the work are given in section 4.6.

4.2 Observations

This section includes a number of important observations we have made regarding how the allocation, binding, and sizing of clock mesh resources (mesh segments, stub wires, and mesh buffers) affect the clock skew.

Observation 1 (*Mesh segment allocation and stub binding*): Some mesh segments in a clock mesh can be safely removed (i.e. deallocated) if the clock skew tolerance constraints are still met without the help of the segments. However, we have observed that the decision of allocating or deallocating a mesh segment is not a simple matter and involves a lot of complication. Precisely, the removal of a segment requires the clock sink whose stub wire has been connected to the segment to be reconnected to another nearby segment in the clock mesh. Thus, the mesh segment allocation or deallocation with stub rebinding may cause to unbalance the load capacitance of mesh driving buffers as well as likely to enlarge the difference of stub wire lengths, thereby increasing the clock skew. Fig. 4.1 shows how the clock skew varies as the (best known) mesh segment optimization algorithm in [4] is applied to benchmark circuit S13207 [2]. The x-coordinate indicates the sequence of mesh segment removals with stub rebinding the algorithm performed, and the y-coordinate indicates the resulting value of clock skew. It is shown that there are mesh segment removals, around the 33rd, 100th, and 150th segment removals, in the sequence that cause a drastic increase of clock skew. This observation clearly implies that a precise stopping criterion or prediction is necessary during the mesh segment allocations or deallocations.

Observation 2 (*Sequence of mesh segment allocations*): The work in [4] used the network sensitivity theory to compute, for each trial of mesh segment deallocation,

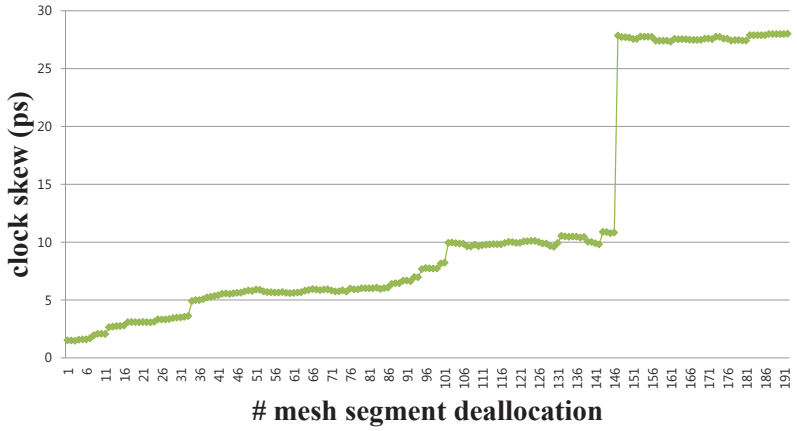
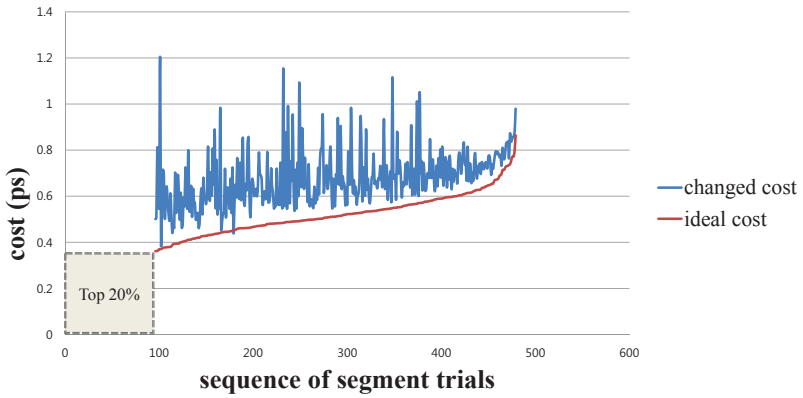


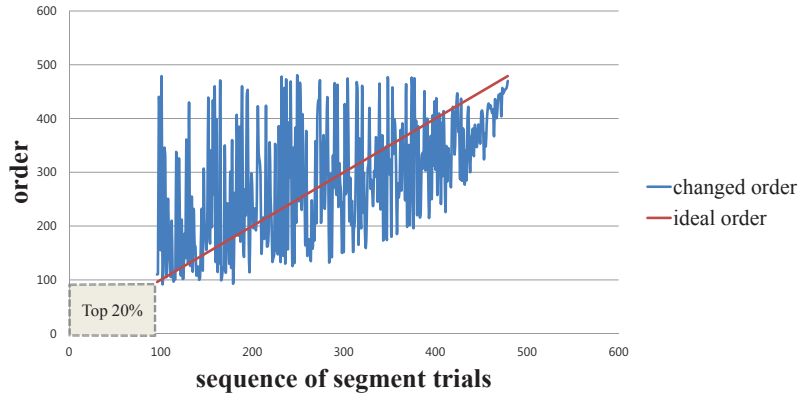
Figure 4.1: The change of clock skew for a sequence of mesh segment deallocations with stub rebinding by [4]. A drastic increase of clock skew has been observed around the 33rd, 100th, and 150th segment deallocations.

a cost to measure how much the deallocation adversely affects the clock skew, and arranged the deallocations in a sequence according to their costs, as shown in the x-coordinate of Fig. 4.1. After performing the deallocations of top 20% on the sequence in Fig. 4.1, we recompute the cost of the remaining 80% using the network sensitivity theory in [4]. The blue lines in Fig. 4.2(a) show the cost change. If the network sensitivity theory used were perfectly accurate, there should be no cost change and the cost should follow the red line in Fig. 4.2(a). Since it is seen that there is a substantial cost change, the deallocation sequence order should be changed accordingly. The blue lines in Fig. 4.2(b) show the order changes. This observation incites that a more comprehensive cost (re)computation that takes into account the effect of a mesh segment allocation/deallocation on the next trials of segment allocations or deallocations is required.

Observation 3 (*Mesh driving buffer sizing*): During the allocation/deallocation of mesh segments with stub rebinding, the load capacitance on the nearby mesh buffers



(a) Cost change of the remaining sequence of the segment deallocation.



(b) Order change of the remaining sequence of the segment deallocation.

Figure 4.2: The changes of sensitivity cost and sequence order of the remaining (80%) segment deallocation trials updated after performing top 20% on the sequence of mesh segment deallocation trials in the x-coordinate of Fig. 4.1.

will vary. Thus, mesh buffer resizing is needed. In the prior works, the mesh buffer sizing and mesh segment allocation are performed separately to reduce the run time complexity. However, since the mesh buffer sizing also drastically influences the clock signal delay, the task of buffer sizing should be combined with the task of mesh segment allocation/deallocation to find a clock mesh network with globally optimized mesh resources.

Motivated by the observations, we propose an integrated clock mesh synthesis algorithm that iteratively performs the three tasks of mesh segment allocation, mesh buffer sizing, and stub binding simultaneously, in which the essential parts are pursuing the computation efficiency as well as the accuracy of the cost that guides the iterations.

4.3 Framework of Clock Mesh Optimization

Our proposed framework of an integrated clock mesh optimization, called CMESH-int, follows the iterative optimization flow of the well-known Kernigan-Lin (K-L) partitioning algorithm [47]. Fig. 4.3 shows the flow diagram of CMESH-int. (Note that our framework is flexible enough to accommodate any iteration flow only if the iteration corresponds to the local incremental updating or refinement and nothing else such as simulated annealing.)

The inputs of CMESH-int are an initial clock mesh, a mesh size of $n \times m$, a buffer library, and a clock skew tolerance constraints. Any clock mesh that satisfies the clock skew tolerance constraints can be used as an initial clock mesh. In this work, we use, as an initial clock mesh, the fully connected clock mesh of size $n \times m$ with the clock sinks bound to the closest mesh segments and the sizing of buffers obtained by the method in [4]. Then, the problem we want to solve can be formally described as:

Problem 1. Clock mesh resource optimization: *Given a buffer library, clock skew*

tolerance constraints[1]*, and a mesh size of* $n \times m$*, generate a clock mesh structure* $\mathcal{M}$ *by (1) allocating/deallocating mesh segments, (2) binding clock sinks to segments (i.e., binding stubs), and (3) sizing mesh buffers with the objective of minimizing the quantity of* C*:*

$$C(\mathcal{M}) = \alpha_1 \sum_{\forall s_i \in \mathcal{S}} l(s_i) + \alpha_2 \sum_{\forall u_i \in \mathcal{U}} l(u_i) + \alpha_3 \sum_{\forall b_i \in \mathcal{B}} area(b_i) \tag{4.1}$$

while satisfying the clock skew tolerance constraints, where $\mathcal{S}, \mathcal{U}$*, and* $\mathcal{B}$ *are the sets of mesh segments, stubs, and buffers in* $\mathcal{M}$*, and* $l(s_i)$*,* $l(u_i)$*, and* $area(b_i)$ *are the length of mesh segment* s_i*, the length of stub* u_i*, and the area of buffer* b_i*, respectively.* α_1*,* α_2*, and* α_3 *are the weighting factors.*

CMESH-int consists of two loops, one loop nesting the other. At each iteration of the *inner-loop*, CMESH-int extracts all feasible moves of incremental movements (i.e., reallocation/removal of mesh segments, stub rebinding, and buffer resizing, which are precisely described in section 4.3.1) and selects a "best move" and performs the movement. Then, the mesh segment resources involved in the movement are frozen and the iterations repeat until there is no feasible[2] move. Then, in the *outer-loop*, among all the feasible mesh structures produced during the iterations of the *inner-loop*, the one (say $\mathcal{M}$) which has the smallest value of C in *Eq.4.1* is selected. All resources in $\mathcal{M}$ are then unfrozen and the *inner-loop* starts again from $\mathcal{M}$. The *outer-loop* stops when there is no reduction on the value of C. The cost function (ΔC) which will be used to select the best move in the *inner-loop* is defined as:

$$\Delta C(f_i) = \alpha_1 \Delta S_{tot} + \alpha_2 \Delta U_{tot} + \alpha_3 \Delta B_{tot} \tag{4.2}$$

where ΔS_{tot}, ΔU_{tot}, and ΔB_{tot} represent the amounts of change of total length of

[1]The clock skew tolerance constraints are described in section 4.3.2.

[2]A clock mesh structure or a resource update is called *feasible* if the structure or update satisfies the clock skew tolerance constraints.

mesh segment, total length of stubs, and total area of buffers resulting from the application of a feasible movement f_i, respectively.

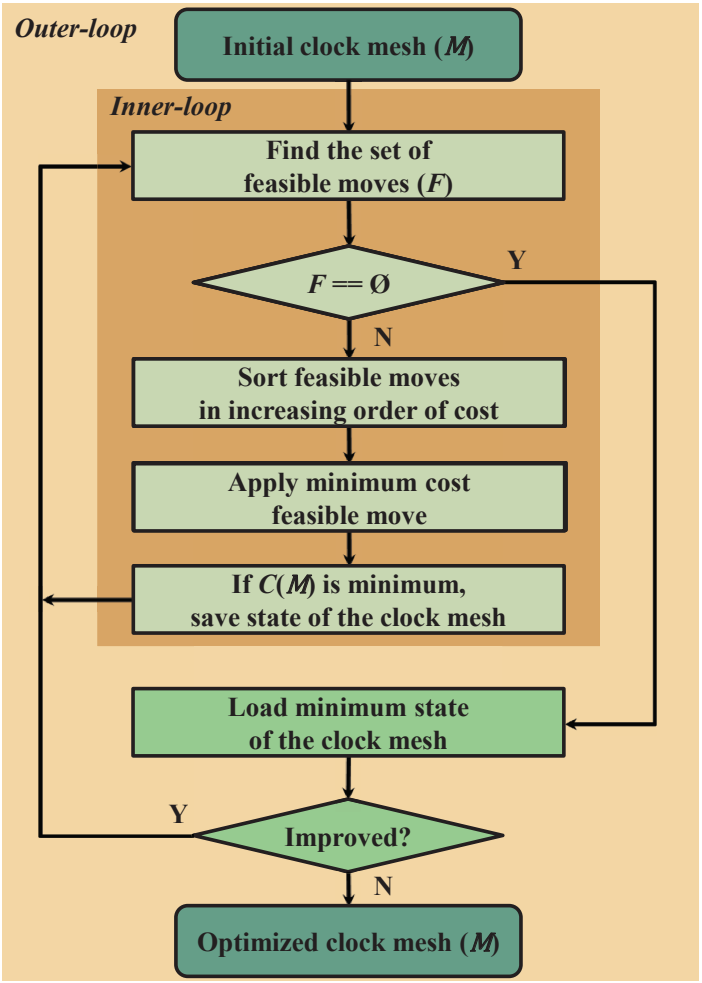


Figure 4.3: The flow diagram of our proposed CMESH-int for clock mesh resource optimization.

The next two subsections define the set of incremental updates of clock mesh resources by CMESH-int and the clock skew tolerance constraints to be met.

4.3.1 Incremental Resource Updates

The clock mesh resource updates are categorized into three types.

1. *Allocate + Upsize* : It corresponds to allocating a mesh segment that was deallocated before followed by a possible upsizing of the nearby mesh buffers to meet the clock slew constraint due to the increased load capacitance.

   To resize buffers, we make use of the result produced by the mesh node covering method in [3]. The mesh node covering method used in our context works as follows. We set the capacitance of each node in a clock mesh to the sum of the capacitances of the mesh segments, stubs, and clock sinks that are closer to the node than any other node. Thus, the total sum of the capacitances of all nodes in the mesh is exactly the total capacitance of the clock mesh. The covering region of a mesh buffer b_i at a mesh node is defined to be the maximal set of nodes around b_i such that the total capacitance of the nodes in the set is not larger than the maximum load capacitance of b_i. We apply the buffer resizing method in [4] to the best mesh structure found so far in the *outer-loop* of CMESH-int to resize the buffers. If the result of refreshed buffer resizing does not violate the variation tolerance constraints (i.e., feasible), we accept the sizing and its node covers as inputs to the *inner-loop*. Otherwise, we simply use the best mesh structure found so far in the *inner-loop*. For the local updates in the *inner-loop* of CMESH-int, the covers that exceed their maximum load capacitances are extracted and upsizing is applied to those buffers in the covers. Thus, the local updates in the *inner-loop* as well as the global updates in the *outer-loop* both will meet the slew constraint. Fig. 4.4 shows an example of segment allocation with buffer upsizing where the segment marked with red color is inserted, resulting in upsizing the buffer in the upper cover of mesh nodes to meet the clock slew constraint.

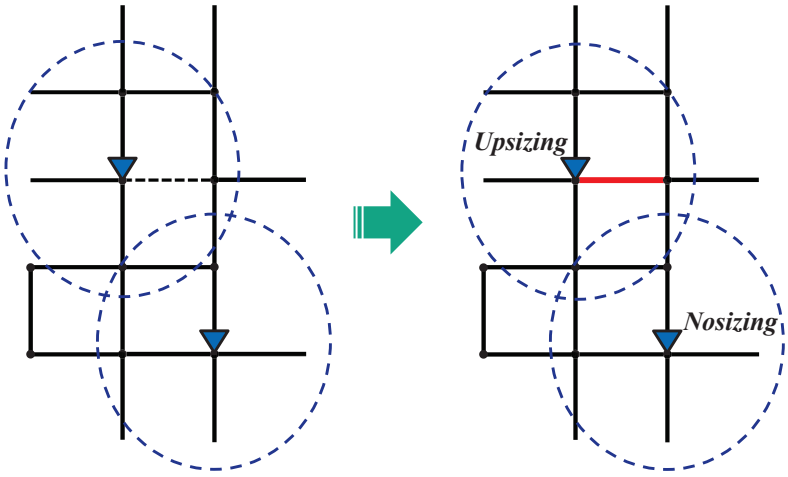


Figure 4.4: Illustrative examples of the clock mesh resource updates (*Allocate + upsize*). A mesh segment (red color) is allocated, resulting in upsizing the buffer in the upper cover of mesh nodes.

2. *Deallocate + Downsize* : It corresponds to deallocating a mesh segment to which no stub is connected. Since the deletion of a mesh segment decreases the load capacitance of the mesh, some of the mesh driving buffers need to be downsized. As the case of *allocate + upsize* does, downsizing of buffers based on the node covering concept will be applied. Fig. 4.5 shows an example of segment deallocation with buffer downsizing where the segment marked with red color is deleted, resulting in downsizing the buffer in the upper cover of mesh nodes while the clock slew constraint is still met.

3. *Deallocate + Size + Bind* : It corresponds to deallocating a mesh segment to which at least one stub is connected. Unlike the case of *Deallocate + Downsize*, it requires to rebind each stub connected to a segment, which will be deallocated, to a nearby segment. That is, the segment selection problem for stub rebinding

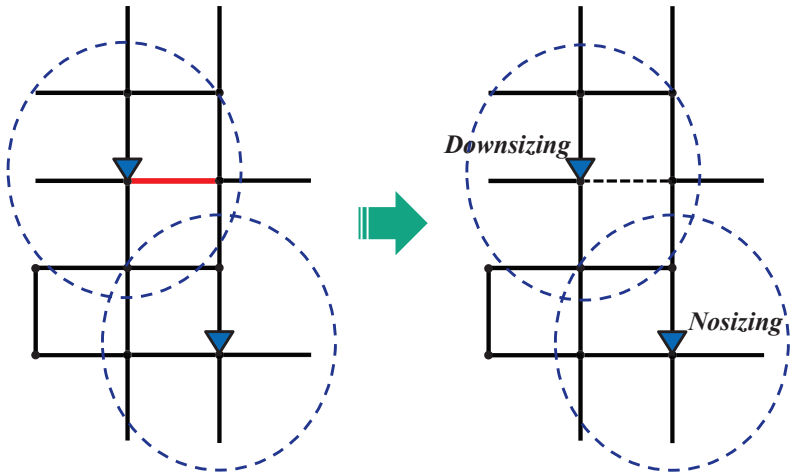


Figure 4.5: Illustrative examples of the clock mesh resource updates (*Deallocate + downsize*). A mesh segment (red color) is deallocated, resulting in downsizing the buffer in the upper cover of mesh nodes.

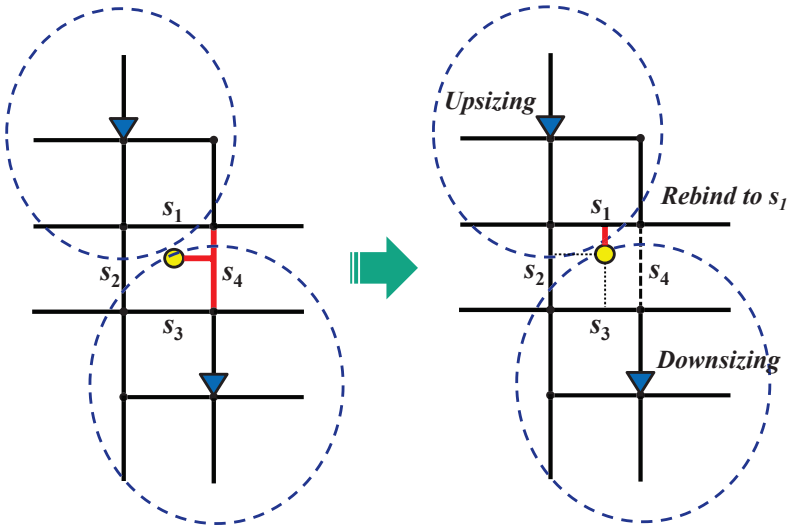


Figure 4.6: Illustrative examples of the clock mesh resource updates (*Deallocate* + *Resize* + *Bind*). A stub (red color) is rebound to segment s_1, resulting in upsizing the buffer in the upper cover of mesh nodes and downsizing the buffer in the lower cover.

needs to be solved. Once the segment to which the stub is to be rebound is determined, buffer downsizing or upsizing will be performed in the same manner as the node covering based procedure used in *Allocate + Upsize* and *Deallocate + Downsize*. Since the segment selection will change not only the stub length but also the clock skew, a careful segment selection is required. We propose a *path-length aware stub rebinding method*, which performs two steps.

Definition 1. Path length $L(u_i)$ of stub u_i: *The path length $L(u_i)$ of stub u_i that is bound to a segment is defined to the sum of the length of the stub and the shortest path length between the location of the segment to which the stub is bound and the buffer that is nearest the segment.*

• **Step 1** (*Generating alternative stub rebindings*): This step is to enumerate all the possible alternative segments to be which the stub is to be rebound. For example, in Fig. 4.6, for the stub that was connected to segment s_4, its alternatives are the closest upward, downward, and left segments. Suppose that stub u_i is to be rebound. For each rebinding of u_i, we compute the path length $L(u_i)$. Let $R(u_i)$ be the set of the path lengths of the rebindings of u_i. Further, let $l_{min}(u_i)$ and $l_{max}(u_i)$ be the minimum and maximum values in $R(u_i)$, respectively.

• **Step 2** (*Selecting a stub rebinding with clock skew awareness*): Let $\mathcal{U}$ be the set of all stubs in the mesh. We set L_{min} to the value of $min\{L(u_k), u_k \in \mathcal{U} - \{u_i\}\}$ and L_{max} to the value of $max\{L(u_k), u_k \in \mathcal{U} - \{u_i\}\}$. Let $R'(u_i)$ be the subset of $R(u_i)$ such that any value in $R'(u_i)$ is in between L_{min} and L_{max}. Then, we check set $R'(u_i)$. If $R'(u_i) \neq \phi$, we choose the rebinding of the shortest stub length among the rebindings corresponding to the values in $R'(u_i)$. This is because $R'(u_i) \neq \phi$ means that the clock skew is not likely to change only if the rebinding selection is taken from $R'(u_i)$ and we want to minimize the wire overhead. Otherwise, we choose the rebinding corresponding to the value in

$R(u_i)$ that is closest in distance from the end points of the interval $[L_{min}, L_{max}]$, which is intended to minimize the increase of the clock skew by stub rebinding. For example, in Fig. 4.6, the stub is decided to be rebound to segment s_1, which results in upsizing the buffer in the upper cover because of the increased wire load capacitance on the cover while downsizing the buffer in the lower cover because of the decreased wire load capacitance on the cover. Fig. 4.7 shows the flow diagram of the two-step stub rebinding algorithm.

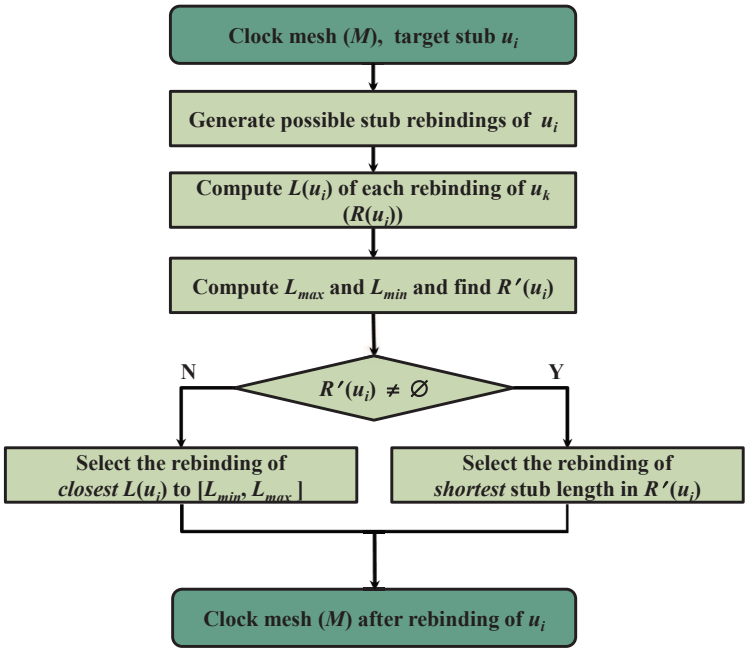


Figure 4.7: The flow diagram of the path-length aware stub rebinding in CMESH-int.

4.3.2 Constraints for Variation Tolerance

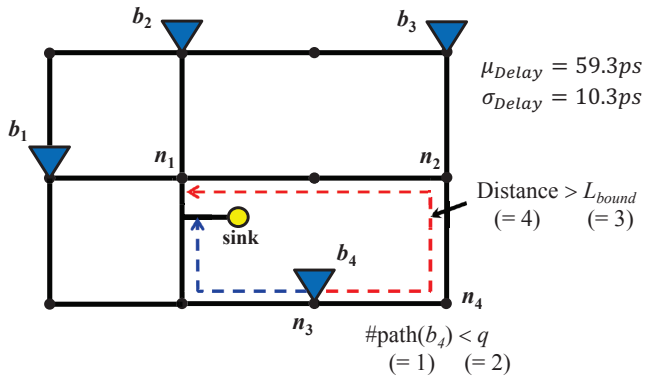
During the incremental resource updates in CMESH-int, besides satisfying the clock skew constraint, it is very important for the updated clock mesh to ensure the delay variation tolerance. Various parameters can be included in the set of variation tolerance

constraints, such as PVT (process-voltage-temperature) variation, noise, and electrical migration parameters. In our work, we include the following two constraints to be met to guarantee variation tolerance of the clock mesh during every optimization stage of CMESH-int.

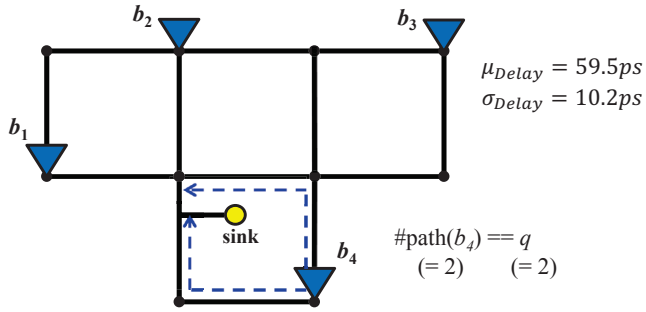
1. *Bounded clock skew constraint*: Under the use of nominal delay values, satisfying the bounded clock skew constraint is likely to lead to meet the clock skew constraint under the delay variation. Moreover, the constraint avoids generating mesh structures of highly unbalanced capacitance loads on mesh buffers and unacceptably long stub wires. Note that many works in the clock tree synthesis [17, 30] have supported bounded clock skew constraint to allow a reasonable range of clock skew while minimizing the resource overhead. However unlike the clock tree, measuring the clock skew in the structure of clock mesh is not as simple as that of the clock tree. Our acceleration techniques for the clock skew estimation will be described in section 4.4.

2. *Multiple path constraint*: Intuitively, if there are many, possibly disjoint, paths to a sink from its driving buffers, the delay variation tolerance to the sink will increase. Thus, maintaining the number of paths to every clock sink from its driving buffers over a certain threshold can be a good constraint in order to ensure the delay variation tolerance.

   One noticeable work regarding the multiple path constraint is that formulated by Venkataraman, Feng, and Li in [3], in which they suggested *network survivability* based clock mesh optimization. They constrained every clock sink to have at least q edge disjoint paths from *each* of at least p closest driving buffers whose distance, in terms of the number of segments, is within L_{bound}. (Parameters p, q, and L_{bound} are empirically determined by user.) For example, Fig. 4.8 shows two partial structures of clock mesh, in which each has one clock sink (yellow



(a) Not meet the constraint of $(p, q, L_{bound}) = (4, 2, 3)$ in [3] because the number (= 1) of disjoint paths whose lengths are within L_{bound} (= 3) from buffer b_4 to the sink is smaller than q (= 2), even though the number (= 4) of buffers within distance L_{bound} is not smaller than p (= 4). For the new formulation, N_{buf} = 4 and N_{seg} = 13, which corresponds to all segments except the two segments (n_2, n_4) and (n_3, n_4).



(b) Meet the constraint of $(p, q, L_{bound}) = (4, 2, 3)$ in [3] because the number of disjoint paths whose lengths are within L_{bound} from each of the four buffer to the sink is not smaller than q (= 2) as well as the number of buffers within L_{bound} is not smaller than p (= 4). For the new formulation, N_{buf} = 4 and N_{seg} = 13, which corresponds to all segments.

Figure 4.8: Two clock mesh structures of almost the same level of variance tolerance. However, the conventional formulation of multiple path constraint considers the two structures having different degrees of variance tolerance, while the new formulation considers them having the same degrees of variance tolerance. (a) The structure on which the conventional multiple path constraint [3] is not satisfied. (b) The structure on which the conventional multiple path constraint [3] is satisfied.

color)[3] and 4 driving buffers. When (p, q, L_{bound}) = $(4, 2, 3)$, the mesh structure in Fig. 4.8(a) does not satisfy the constraint since there exists only one path from b_4 to sink of distance less than or equal to L_{bound} (= 3). On the other hand, the structure in Fig. 4.8(b) satisfies the multiple path constraint. To assess the effectiveness of the constraint formulation of (p, q, L_{bound}) further, we applied a Monte-Carlo simulation using Hspice to the mesh structures in Fig. 4.8 and measured the average delay and standard deviation. We see that the average delays and standard deviations are almost the same. In addition, we measured that the worst delay difference was $0.1ps$. From the simulation, it is observed that the two mesh structures in Fig. 4.8 entail very similar variation tolerances. This means that the conventional multiple path constraint of (p, q, L_{bound}) is not well suited for the variation tolerance. One main reason is due to the tight constraint of requiring at least q paths from *each* buffer to sink. The constraint ignores the case where some buffers have a small number of paths while other buffers have a large number of paths, thus overall, the total number of paths is sufficient to ensure the variation tolerance.

Based on the above analysis, we propose a new formulation of the multiple path constraint which is simpler but stronger than the constraint (p, q, L_{bound}) in [3]. We compute $\rho(u_i)$ of a stub u_i called *the degree of collective path redundancy*:

$$\rho(u_i) = N_{buf}(u_i) + w \times N_{seg}(u_i) \tag{4.3}$$

where $N_{buf}(u_i)$ and $N_{seg}(u_i)$ represent the number of driving buffers whose distance is less than or equal to L_{bound} to the sink on u_i and the total number of (disjoint) mesh segments on the paths of length less than or equal to L_{bound} from the buffers counted in N_{buf} to the sink on u_i, respectively. (Note that the

[3]For the path length computation, the sink is assumed to be connected to the closest node n_1 in the mesh in Fig. 4.8.

two structures in Fig. 4.8 have the same values of N_{buf} and N_{seg}, which are 4 and 13, respectively.) We set the weight factor w to $\ll 0.1$ since mesh buffers are driven by the top-level of clock tree and thus is able to resolve greater level of variation tolerance than the mesh segments. We maintain the value of $\rho(\cdot)$ for every stub. If the $\rho(\cdot)$ value is set to a large (or small) value, a robust (or weak) variation tolerance is maintained at the expense (or save) of mesh resources. In our experiments, we found that $\rho = 3 \sim 4$ leads to an excellent trade between variation tolerance and resource reduction.

4.3.3 Early Decision Policies

Even though CMESH-int employs a set of acceleration techniques to speed up the estimation of clock skew, it could suffer a long run time if it performs *all iterations* and evaluates the feasibility of *all resource updates* at each iteration. We propose two early decision policies that are able to greatly save computational effort at the expense of a slight quality loss of results.

- *Cut-down the number of incremental update moves*: Note that at each iteration of the *inner-loop*, the feasibility test for every incremental update is attempted and the incremental update which has the least value of $\Delta C(\cdot)$ in *Eq.4.2* is selected for the resource update. The feasibility test refers to checking the two constraints: bounded clock skew constraint and multiple path constraint. For the multiple path constraint, finding shortest paths on a local region of mesh is fairly simple and fast. However, for the clock skew constraint, it involves relatively complicated matrix computations. Based on the analysis, our feasibility test policy performs in four steps:

  1. The multiple path constraint is checked for every move of incremental update.

2. The moves that satisfy the multiple path constraint is arranged in nondecreasing order according to the $\Delta C(\cdot)$ value in *Eq.4.2*.
3. The clock skew constraint is checked for only the top $\gamma\%$ of the moves on the list. (The γ value can be determined empirically. In our work, it is set to 50.)
4. The move with the least $\Delta C(\cdot)$ value in *Eq.4.2* is chosen as the update.

- *Early termination of iterations*: Ideally, CMESH-int runs the *inner-loop* until no feasible move is found. However, we have observed that CMESH-int spends unnecessary effort for a large part of iterations. Our practical policy is that if $\Delta C(\cdot)$ value of the chosen move for incremental update in each iteration exceeds a certain threshold, we stop the iteration. By carefully setting the threshold value, the computational effort can be saved significantly.

4.3.4 Time Complexity Analysis

At each iteration of the *inner-loop*, CMESH-int checks the feasibility of possible moves. The clock skew calculation of possible moves is the most time consuming part of the feasibility check. The clock skew of the clock mesh is calculated in two steps. The time complexity of LU decomposition of the clock mesh is $O((n \cdot m + |U|)^{\kappa})$ where n and m are the number of horizontal and vertical nodes, $|U|$ is the number of stubs and $1 \leq \kappa \leq 3$ which depends on the number of nonzero elements and sparsity patten of matrix. The number of possible moves is $|S|$ in the worst case where $|S|$ is the number of mesh segments. Hence, the time complexity of finding the best move is $O(|S| \cdot ((n \cdot m + |U|)^{\kappa}))$. Since CMESH-int tries to find a best move until there's no feasible move, every segment can be a best move in the worst case. Therefore, the time complexity of an iteration of the *outer-loop* is $O(|S|^2 \cdot ((n \cdot m + |U|)^{\kappa}))$. CMESH-int needs $2 \sim 7$ iterations of the *outer-loop* to optimize the clock mesh, depending

on variation tolerance constraints and benchmark circuits in our experiments. Furthermore, we reduce the time complexity by using acceleration techniques presented in the next section. Assuming that the clock mesh is partitioned into $|W|$ sliding windows, the number of nodes of RC-network of a sliding window is approximately bounded by p ($\ll n \cdot m + |U|$). Furthermore, by partially reusing the result of LU decomposition, it needs 2 LU compositions for clock skew calculation of possible moves in each sliding window. Therefore, the time complexity of accelerated CMESH-int is bounded by $O(|S| \cdot |W| \cdot (p^{\kappa}))$.

4.4 Fast Clock Skew Estimation Techniques

This section describes a set of fast clock skew estimation techniques in a full or partial structure of clock mesh. First, we discuss the details on the clock signal delay calculation in a clock mesh. Then, in the first subsection, by modifying the RC circuit model, we propose a method of partial reusing the computation of matrix multiplications for fast clock skew estimation. In the second and third subsections, we discuss how the sliding window scheme can be used in the clock skew estimation and how quickly estimate the effect of buffer sizing on the clock skew, respectively. Finally, the last subsection discusses the effectiveness of the three acceleration methods in terms of accuracy and run time.

• *Clock signal delay calculation on a clock mesh*: The Elmore delay of RC network is defined by the first moment of the impulse response. In RC-tree structures, Elmore delay can be calculated simply by traversing the RC-trees. However, Elmore delay of general RC networks is calculated by matrix computation because of the presence of resistor loops in the RC networks. The first moment of the impulse response is calculated by solving the dc-equivalent circuit of RC network [48], which refers that each capacitor is replaced by a current source with the capacitance as its current value.

In dc-equivalent circuit, the voltage of each node $x_i \in X$ represents its Elmore delay such that $G * X = C$, where G is the $N \times N$ conductance matrix, C is the $N \times 1$ capacitance matrix, and X is the $N \times 1$ node voltage vector. The value of X is obtained by solving $G * X = C$, which is solved by two steps: (1) LU decomposition on G and (2) finding X using the matrices L and U in (1) and C. Generally, the LU decomposition takes much longer time than solving X using the matrices L and U. Consequently, if we are able to reuse the result of LU decomposition, the overall computation time could be significantly reduced.

4.4.1 Partially Reusing Matrix Multiplication for Incremental Updates

We modify the RC circuit model to enable a fast clock skew estimation for the incremental updates. We model wires of the clock mesh by 1-π model in which a wire segment is modeled with one resistor in the middle of the segment and two capacitors at the end points of the segment. Fig. 4.9 shows an example of 1-π model of RC circuit, in which for the deallocated mesh segments (red color) we assign an extremely large value, denoted as r_∞, of resistance. The reason for introducing r_∞ is that we want to keep all the nodes in the mesh regardless of the allocation and deallocation status of mesh segments. This will facilitate the reuse of a partial computation of $G * X = C$, as it will be explained in the next paragraph. (We have observed from experiments that the use of our modified RC circuit model changes the clock skew estimation only within 0.1ps.)

For the incremental update of allocating a mesh segment s_i of length $l(s_i)$ and capacitance $cap(s_i)$, the resistor on its edge in the RC circuit model changes from r_∞ to $r_w \cdot l(s_i)$, in which r_w is per unit resistance of segment and the capacitor at each node adjacent to the edge increases by $cap(s_i)/2$ while for the incremental update of deallocating a mesh segment s_i, the resistor changes from $r_w \times l(s_i)$ to r_∞ and the capacitor decreases by $cap(s_i)/2$. Let us consider the case of allocating s_i. Since the insertion

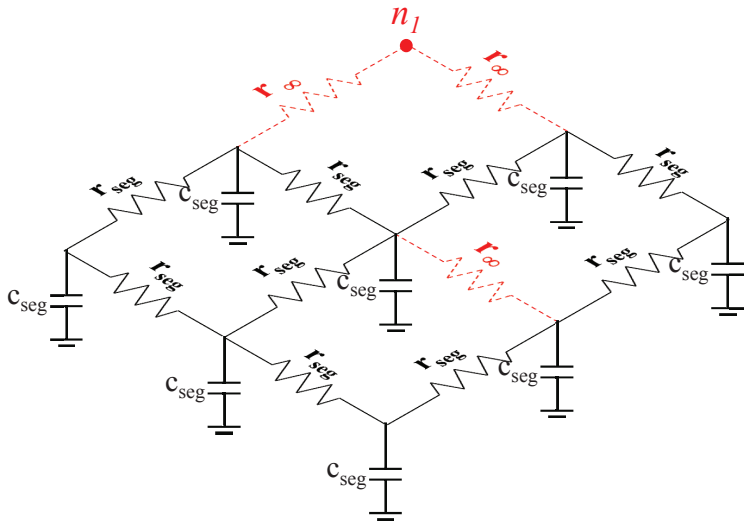


Figure 4.9: Our modified 1-π RC circuit model.

of s_i changes both G and C, theoretically, we should perform LU decomposition and solving X using the L and U for every trial of the incremental allocation. However, as exploited by the link insertion works in [40, 41], the new delay (X') caused by the (parallel-edge) addition of resistance r_{in} to an edge, say (a, b), in an RC circuit is calculated by [49]:

$$X' = X - \frac{x_a - x_b}{r_{in} + (r_a - r_b)} \cdot R \tag{4.4}$$

where X represents the delay matrix of the original RC circuit, r_a and r_b indicate the transfer resistances of nodes a and b, and R represents the transfer resistance matrix.

In our context of allocating a segment s_i, r_{in} is computed such that $1/r_{in}$ = $1/(r_w \cdot l(s_i))$ - $1/r_\infty$ and (a, b) corresponds to the edge of s_i in the RC network. We can compute X' in *Eq.4.4* by performing the following four steps:

1. Update the capacitance matrix by adding $cap(s_i)/2$ to each of a and b. Let C' denote the updated capacitance matrix. Compute X by solving $G * X = C'$.

2. Create an auxiliary RC circuit from the original RC circuit by deactivating all independent sources and including a unit current source between nodes a and b.

3. Compute transfer matrix R, transfer resistances r_a and r_b by solving the auxiliary RC circuit obtained in Step 2.

4. Compute X' in *Eq.4.4* using X, R, r_a, and r_b obtained in Steps 1 and 3.

Note that Steps 1 and 3 each performs LU decomposition and the decomposition result in each step does not vary, irrespective of allocation of any mesh segment. Consequently, the LU decomposition result in each of Steps 1 and 3 can be totally reused among all the trials of incremental update in the *inner-loop* of CMESH-int. For the incremental update of deallocating a segment, the computation procedure of X' is exactly identical except a slight change of the value of C' in Step 1 and the expression in *Eq.4.4*.

4.4.2 Adopting Sliding Window Scheme

Although we can reuse the result of LU decompositions, the mesh size and the number of stubs greatly affect the time required for checking the feasibility of each incremental update. As validated in [5], the sliding window scheme on clock mesh can greatly reduce the run time to analyze the clock skew under a local change on the mesh. In the sliding window scheme in [5], the clock mesh of size $n \times m$ is divided into a disjoints set of windows (W) of each size $r \times s$. (The values of r and s are chosen to multiple integers of n and m.) and a *border* $B(W)$ of a W is defined to the window of size $p \times q$ ($r \leq p$, $s \leq q$) that properly contains W. An example of the partition of a mesh into sliding windows by [5] is shown in Fig. 4.10(a).

Once the partition is done, the mesh segments, stub wires, and clock sinks in each window W are expressed in detail and the delay to the clock sinks is calculated. On the

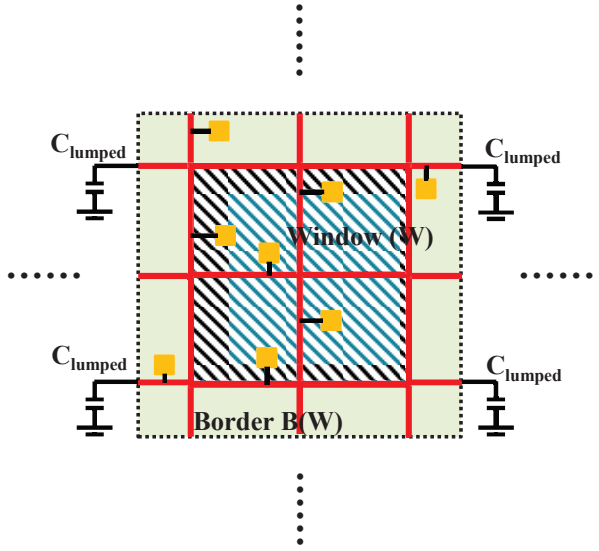
other hand, even though the resources in its border $B(W)$ are expressed in detail, the delay to the sinks in $B(W)$ is not calculated, and the rest part of clock mesh beyond $B(W)$ is replaced by the lumped capacitance. When the signal delays on a window are to be calculated, the signal delays in the other windows are treated as constant.

4.4.3 Adjusting Delay Caused by Buffer Resizing

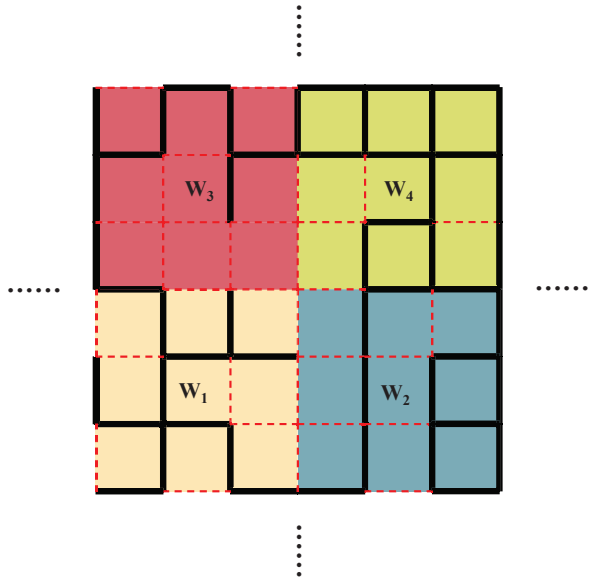
Since CMESH-int resizes the mesh driving buffers due to the capacitance change by the incremental allocation and deallocation of mesh segments, the delay at each node (i.e. $X' = [x'_1, x'_2, \cdots, x'_N]$ in *Eq.4.4*) in the RC network should be updated accordingly. We use a similar approach to that used in [50] which analyzes how the IR-drop changes as the pad location is updated. Specifically, given an RC circuit model, we are able to compute the Elmore delay of a node n_i by iteratively applying the Kirchoff's Current Law shown by *Eq.4.5* and *Eq.4.6*, in which ADJ_i is the set of adjacent nodes of n_i, $g_{i,j}$ is the conductance between nodes n_i and n_j, $cap(n_i)$ is the capacitance on n_i, and λ is a weighting factor. If the delay change at n_i is larger than a given value, which is typically in $0.01 \sim 0.1ps$, the nodes in ADJ_i are added to the set A of influenced nodes. Then, the process iterates with A. The iterations stop if the delay change at every node in A is less than a threshold ϵ. Thus, CMESH-int can trade accuracy with run time by controlling the value of ϵ.

$$x'_i = \frac{\sum_{n_j \in ADJ_i} g_{i,j} x'_j}{\sum_{j \in ADJ_i} g_{i,j}} - \frac{cap(n_i)}{\sum_{n_j \in ADJ_i} g_{i,j}}, \tag{4.5}$$

$$x'^{k+1}_i = \lambda \cdot x'^{k+1}_i + (1 - \lambda) \cdot x'^{k}_i, \;\; k = 1, 2, \cdots . \tag{4.6}$$



(a) A single sliding window



(b) The sliding windows when $r = s = 4$

Figure 4.10: An example using sliding window scheme. (a) Sliding windows in [5]. (b) An example of sliding windows when $r = s = 4$.)

4.5 Experimental Results

4.5.1 Experimental Environments

Our proposed integrated approach CMESH-int for clock mesh resource optimization under variation tolerance constraints has been implemented in C++ and run on Linux machine with 2.5 GHz Intel Xeon processor and 24GB memory. We used UMFPACK [51] for the LU factorization in our clock skew analysis as well as in the sensitivity analysis of MeshWorks [4]. We use 12 buffers of different sizes with the maximum capacitance ranging from 60fF to 500fF under a slew constraint of 100ps, which is 10% of 1GHz clock frequency. For benchmark circuits of [6], we use 4 additional buffers that can drive from 600fF to 900fF, since benchmark circuits in [6] have larger clock sink density and clock sink capacitance than [2]. The technology parameters and transistor models are based on 45nm predictive technology model (PTM) [52]. The per unit resistance and capacitance are $r_w = 0.1\Omega/\mu m$ and $c_w = 0.2fF/\mu m$.

We consider four variation parameters: power supply voltage, buffer channel length, wire width, and sink capacitance. Each variation parameter is assumed to be a normally distributed random variable whose value varies with 5% standard deviation around its nominal value, in which each variation parameter is varied by $\pm 15\%$ in the worst case, generally meaning 3σ. We also consider a spatial correlation on each parameter for which we apply PCA (principal component analysis) [53]. We have modeled the variations at the top-level tree of clock mesh in the same manner as that used in [4]. The clock skew between mesh buffers (arrival time) is modeled as a random variable within the range of $\pm 50ps$ which means that worst case clock skew of input signal is $100ps$ and the clock slew is modeled as a random variable within the range of $100 \pm 10ps$. We took the placement of the clock sinks of benchmark circuits from [2], which produced the results by applying the Berkeley SIS tool followed by the UCLA Dragon tool, and five benchmark circuits from [6]. Table 4.1 summarizes the initial clock mesh configu-

Table 4.1: Initial clock mesh configurations for benchmark circuits [2] and [6].

Circuit	$\|S\|$/Size	μ_{sk}/σ_{sk}	$skew_{worst}$	WL	BA	PWR
		(ps)	(ps)	(mm)	(μm^2)	(mW)
s9234	212/9x9	4.53/1.87	10.14	41.1	163.4	7.89
s13207	639/13x13	7.46/2.86	16.04	93.3	359.0	17.99
s15850	535/13x13	8.96/3.29	18.83	107.0	391.1	20.45
s35932	1729/21x21	16.01/5.85	33.56	285.8	1041.8	54.88
s38417	1637/21x21	14.55/5.48	30.99	240.6	936.6	46.46
s38584	1427/21x21	14.96/5.62	31.82	251.5	957.8	48.28
01	1107/25x25	19.92/6.18	38.46	460.5	1769.2	103.47
02	2249/33x21	24.72/6.48	44.16	641.9	2552.0	155.99
03	1200/17x9	8.63/3.35	18.68	50.1	450.6	26.29
04	1845/17x17	8.42/3.10	17.72	84.3	478.5	35.54
05	1016/21x21	7.1/2.65	15.05	121.2	664.5	27.83

rations for the benchmark circuits produced by applying the buffer allocation and sizing algorithm in [4] to the fully connected clock meshes with clock sinks. Each entry in the second column of Table 4.1 indicates the number of clock sinks and the clock mesh size of the corresponding circuit. The next five columns represent the mean and standard deviation of the clock skew and worst case clock skew ($\mu + 3\sigma$) under the variation conditions mentioned above (extracted by performing Hspice Monte-Carlo simulation), the total wirelength (WL), the buffering area (BA) and the power consumption (PWR) of the initial clock mesh of each circuit, respectively. The total wirelength includes the length of mesh segments and the length of stub wires.

4.5.2 Resource Requirement and Variation Tolerance Comparison

To demonstrate the efficiency of CMESH-int over that of MeshWorks [4], we also implemented the buffering and buffer resizing algorithm in [4] and the clock mesh optimization algorithm in [4]. Table 4.2 shows the comparison of the resource requirement results produced by MeshWorks [4] and CMESH-int. We applied (sequentially) the network sensitivity based mesh optimization and then the buffer resizing used in MeshWorks [4] to the initial clock meshes in Table 4.1 to produce optimized results of clock meshes. Furthermore, we perform our iterative clock mesh optimization framework on the same initial clock meshes as that used by MeshWorks to produce the optimized results of CMESH-int. We set the cost coefficients of CMESH-int as $\alpha_1 = \alpha_2 = 0.0067/\mu m$ and $\alpha_3 = 1/\mu m^2$, since we use the same wire model for the mesh segment and stub. We compare total wirelength of clock mesh (WL), mesh buffer area (BA), power consumption (PWR), and run time. Power consumption (PWR) is measured by Hspice simulation. The column labelled as β indicates the clock skew constraint we set for each benchmark circuit. The last row of Table 4.2 shows the average ratio of the result produced by MeshWorks to that by CMESH-int. In summary, CMESH-int uses 13.2%, 10.9%, and 11.0% less wirelength, buffering area, and power

Table 4.2: Resource requirement results produced by MeshWorks [4] and our CMESH-int.

Circuit	MeshWorks [4]				CMESH-int				
	WL	BA	PWR	CPU	β	WL	BA	PWR	CPU
	(mm)	(μm^2)	(mW)	(s)	(ps)	(mm)	(μm^2)	(mW)	(s)
s9234	31.3	131.9	6.05	0.27	3.5	25.4	114.7	4.96	4.86
s13207	74.4	301.8	14.49	2.30	7.0	63.0	263.1	12.31	16.57
s15850	84.0	328.9	16.20	1.77	5.5	69.0	276.9	13.35	14.61
s35932	233.2	900.4	45.16	33.85	5.5	199.2	790.1	38.63	117.14
s38417	189.0	791.9	36.90	30.79	6.0	162.4	697.8	31.88	140.75
s38584	193.9	791.8	37.59	23.80	6.0	165.5	711.2	32.31	89.60
01	218.5	1066.9	58.68	25.45	7.5	199.8	987.6	55.07	147.64
02	395.7	1839.4	110.41	89.75	7.5	384.0	1774.4	108.1	406.97
03	43.5	438.2	25.10	10.72	5.0	35.8	394.3	23.60	45.07
04	91.9	504.4	28.69	25.78	5.0	84.3	478.5	27.24	81.34
05	44.7	301.0	13.48	17.40	7.5	39.1	248.0	12.27	58.04
ratio	1.0	1.0	1.0	.	.	**0.868**	**0.891**	**0.890**	.

consumption compared to that by MeshWorks, respectively. Since CMESH-int is an iterative and integrated mesh resource optimization framework, it takes more computational effort than MeshWorks. However, our acceleration techniques efficiently reduce the computation time of clock skew estimation, resulting in an acceptable run time, which is less than 150 seconds for most designs. Design 02 of ISPD10 is the largest circuit in ISPD10 where the chip area is $91mm^2$. Thus, we set the largest mesh size to that in 02. CMESH-int takes about 400s for design 02. However, the ratio to MeshWorks is similar to the other benchmark circuits, which is about 4.5X.

Table 4.3: Clock skew variation results produced by MeshWorks [4] and CMESH-int under high input variation condition.

Input			MeshWorks [4]			CMESH-int		
$IV_{skew}(ps)$	$IV_{slew}(ps)$	Circuit	$\mu_{sk}(ps)$	$\sigma_{sk}(ps)$	$skew_{worst}(ps)$	$\mu_{sk}(ps)$	$\sigma_{sk}(ps)$	$skew_{worst}(ps)$
± 70	100 ± 20	s9234	30.68	12.25	67.43	30.61	8.14	55.03
		s13207	52.51	16.90	103.21	41.80	9.08	69.04
		s15850	48.05	12.97	86.96	45.36	8.80	71.76
		s35932	59.26	12.90	97.96	51.52	8.58	77.26
		s38417	53.15	13.45	93.50	46.61	8.67	72.62
		s38584	60.92	14.38	104.06	45.93	8.43	71.22
		01	64.51	11.83	100.00	53.49	8.78	79.83
		02	73.55	11.46	107.93	53.28	8.28	78.12
		03	27.17	6.66	47.15	20.19	6.43	39.48
		04	54.65	16.37	103.76	35.52	7.97	56.43
		05	37.24	15.84	84.76	26.57	7.48	49.01
		ratio	1.0	1.0	1.0	**0.804**	**0.651**	**0.731**

Since we model the delay between mesh buffers affected by the top-level tree of clock mesh as a random variable of the clock skew and slew, we conduct the clock skew analysis under the three input variation conditions. Table 4.3, 4.4 and 4.5 show

Table 4.4: Clock skew variation results produced by MeshWorks [4] and CMESH-int under midium input variation condition.

Input			MeshWorks [4]			CMESH-int		
$IV_{skew}(ps)$	$IV_{slew}(ps)$	Circuit	$\mu_{sk}(ps)$	$\sigma_{sk}(ps)$	$skew_{worst}(ps)$	$\mu_{sk}(ps)$	$\sigma_{sk}(ps)$	$skew_{worst}(ps)$
± 60	100 ± 15	s9234	26.75	10.71	58.88	26.44	7.06	47.62
		s13207	45.41	14.44	88.73	36.10	7.70	59.20
		s15850	41.71	12.04	77.83	39.08	7.55	61.73
		s35932	51.37	11.24	85.09	45.39	7.87	69.00
		s38417	46.24	11.52	80.80	41.00	7.79	64.37
		s38584	53.02	12.51	90.55	40.67	7.81	64.10
		01	56.14	10.27	86.95	46.74	7.98	70.68
		02	63.76	10.01	93.79	46.76	7.61	69.59
		03	23.81	5.79	41.18	17.47	5.44	33.79
		04	47.10	14.04	89.22	28.15	6.93	48.94
		05	32.73	13.88	74.37	23.33	6.66	43.31
		ratio	1.0	1.0	1.0	**0.806**	**0.661**	**0.736**

Table 4.5: Clock skew variation results produced by MeshWorks [4] and CMESH-int under low input variation condition.

Input			MeshWorks [4]			CMESH-int		
$IV_{skew}(ps)$	$IV_{slew}(ps)$	Circuit	$\mu_{sk}(ps)$	$\sigma_{sk}(ps)$	$skew_{worst}(ps)$	$\mu_{sk}(ps)$	$\sigma_{sk}(ps)$	$skew_{worst}(ps)$
± 50	100 ± 10	s9234	23.10	9.23	50.79	22.51	6.07	40.72
		s13207	38.51	12.03	74.60	30.71	6.47	50.12
		s15850	35.69	10.18	66.23	33.18	6.47	52.59
		s35932	43.89	9.71	73.02	39.75	7.36	61.83
		s38417	39.72	9.72	68.88	35.89	7.09	57.16
		s38584	45.53	10.69	77.60	35.92	7.30	57.82
		01	48.22	8.87	74.83	40.58	7.33	62.57
		02	54.54	8.63	80.43	40.84	7.21	62.47
		03	20.87	5.11	36.20	15.09	4.55	28.74
		04	39.80	11.78	75.14	24.08	5.98	42.02
		05	28.54	11.96	64.42	20.36	5.92	38.12
		ratio	1.0	1.0	1.0	**0.812**	**0.687**	**0.750**

the clock skew variation results produced by MeshWorks and CMESH-int. Besides the input variations, all other parameters are identical to that in Table 4.2. The first three columns denoted by *Input* indicate the input skew variation, input slew variation, and circuit name, respectively. The mean and standard deviation of the resulting clock skews, denoted by μ_{sk} and σ_{sk}, and the worst clock skew ($Skew_{worst}$) used by MeshWorks and CMESH-int are compared in the table. The mean and standard deviation of the clock skew are measured by Hspice Monte-Carlo simulation under the variation conditions mentioned above. It is shown that CMESH-int reduces the worst clock skew by 26.9%, 26.4% and 25.0%, respectively under three input variation conditions.

Table 4.6: Power variation results produced by MeshWorks [4] and CMESH-int.

Circuit	MeshWorks [4]						CMESH-int					
	Short Circuit Power			Total Power			Short Circuit Power			Total Power		
	μ	σ	$\mu+3\sigma$	μ	σ	$\mu+3\sigma$	μ	σ	$\mu+3\sigma$	μ	σ	$\mu+3\sigma$
s9234	0.71	0.17	1.22	6.42	0.67	8.43	0.58	0.14	1.00	5.24	0.55	6.89
s13207	1.71	0.39	2.88	15.36	1.60	20.16	1.39	0.32	2.35	12.99	1.35	17.04
s15850	1.86	0.42	3.12	17.12	1.78	22.46	1.46	0.34	2.48	14.04	1.47	18.45
s35932	5.17	1.05	8.32	47.86	4.43	61.15	4.23	0.87	6.84	40.79	3.80	52.19
s38417	4.51	0.93	7.30	39.34	3.69	50.41	3.73	0.77	6.04	33.83	3.19	43.40
s38584	4.48	0.89	7.15	40.01	3.65	50.96	3.75	0.78	6.09	34.27	3.18	43.81
01	6.16	1.33	10.15	61.67	5.76	78.95	5.56	1.19	9.13	57.70	5.35	73.75
02	10.63	2.18	17.17	115.16	10.35	146.21	10.44	2.16	16.92	112.86	10.13	143.25
03	2.66	0.61	4.49	26.41	2.64	34.33	2.51	0.58	4.25	24.89	2.49	32.36
04	3.07	0.71	5.20	30.19	3.11	39.52	2.88	0.67	4.89	28.66	2.95	37.51
05	1.56	0.37	2.67	14.11	1.45	18.46	1.41	0.33	2.40	12.98	1.33	16.97
ratio	1.0	1.0	1.0	1.0	1.0	1.0	**0.870**	**0.878**	**0.873**	**0.889**	**0.891**	**0.889**

Since more than one clock signal path to a clock sink can exist in the mesh structure, the short circuit power is another source of power consumption in the clock mesh.

The short circuit power grows as the input slew and arrival time difference increase. We perform power analysis for the results produced by MeshWorks and ours by using Hspice Monte-Carlo simulation under the same condition done in Table 4.2 and input variation (skew: $\pm 50ps$, slew: $100 \pm 10ps$), and summarize the results in Table 4.6. We compare the mean, standard deviation, and worst case value ($\mu + 3\sigma$) for each of the short circuit power consumption and the total power consumption. The comparison shows that CMESH-int uses 12.7% less worst short circuit power and uses 11.1% less worst total power.

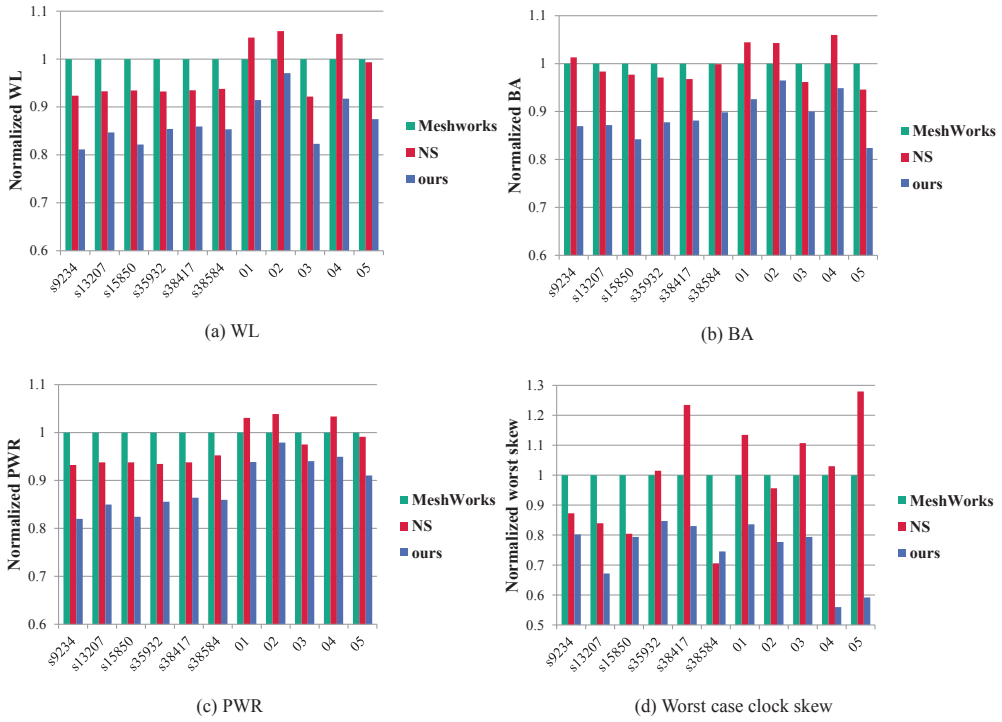


Figure 4.11: Normalized experimental results of MeshWorks, NS (Network Survivability based mesh optimization) and ours

Fig. 4.11 summarizes the results produced by MeshWorks [4] (green), the network survivability based method [3] (red) and CMESH-int (blue). We applied (sequentially)

the mesh optimization in [3] and then the buffer resizing in [4] to the initial clock meshes in Table 4.1 to produce optimized results of the network survivability based method. The worst case clock skew is calculated by Hspice Monte-Carlo simulation under the input variation of skew = $\pm 50ps$ and slew = $100 \pm 10ps$. Each value is normalized with respect to the results of MeshWorks. In summary, ours reduces the worst case clock skew, wirelength (WL), buffer area (BA), and power consumption (PWR) by 25.0%, 13.2%, 10.9%, 11.0% over that of MeshWorks, respectively, and also reduces by 22.4%, 10.5%, 10.6% and 8.6% over that of the network survivability based method, respectively.

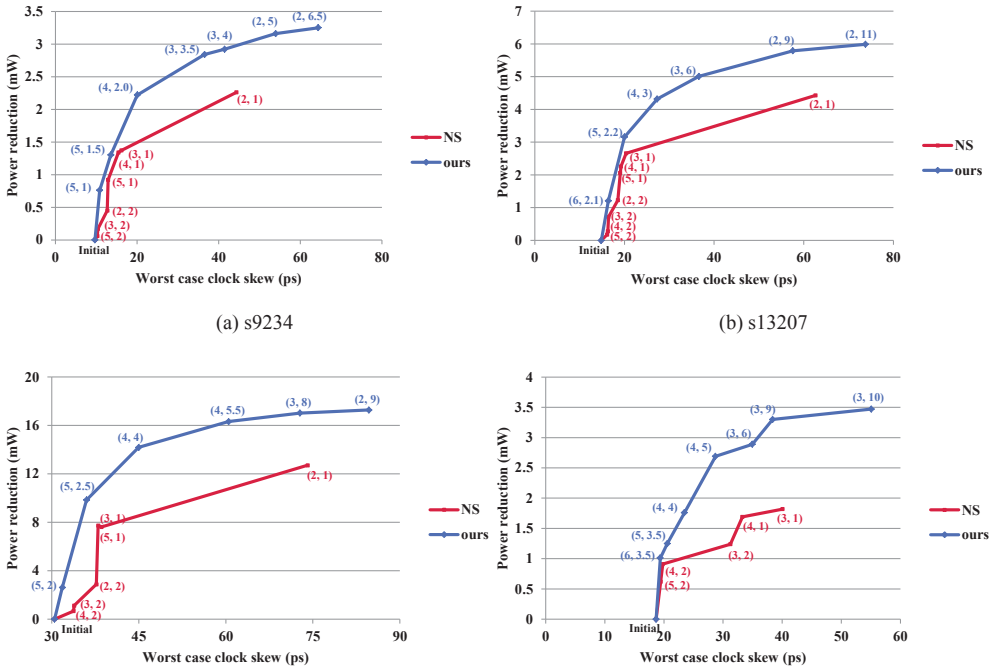


Figure 4.12: Trade-off between worst case clock skew and power reduction of CMESH-int and network survivability based method [3] while varying the constraints of CMESH-int and network survivability based method.

Furthermore, we analyze the trade-off between the worst case clock skew and the power reduction of CMESH-int (blue) and the network survivability based method in [3] (red) by varying the constraints of each method in Fig. 4.12. The x-coordinate indicates the worst case clock skew measured by Hspice Monte-Carlo simulation under the same variation environment used in Fig. 4.11. The y-coordinate indicates the power reduction which is also measured by Hspice simulation. The values on each node of the red curve indicate the network survivability constraint used in [3] in which the constraint values (p, q) means q edge disjoint paths from $each$ of at least p closest driving buffers for every clock sink. On the other hand, the values on each node of the blue curve indicate the multiple path and clock skew constraints used in our CMESH-int. Through the analysis on four benchmark circuits, our proposed method achieves higher power reduction under the same worst case clock skew constraint.

4.5.3 Comparison with Clock Mesh Optimization using Worst Case Timing Analysis of Commercial Tool

We model the clock mesh as the RC network in calculating the clock skew of the clock mesh. Synopsys DCMatch analysis [54] provides the effect of variations in DC response of RC networks. Unlike Monte-Carlo simulation, DCMatch analysis do not rely on the sampling, and is significantly faster than Monte-Carlo simulation with comparatively small error. Since DCMatch analysis calculates the variation of specific output variable or pair of output variables, the runtime of DCMatch analysis is related to the number of output variables to measure the effect of variations. Fig. 4.13 shows the runtime and error between worst case clock skew measured by DCMatch and Monte-Carlo simulation by varying the number of output variables of DCMatch. The x-coordinate indicates the number of sample outputs. The left and right y-coordinates indicate the runtime and error by Hspice DCMatch analysis for s9234 [2]. The number of sinks and random elements of s9234 are 212 and 2078, respectively. We model the variation

parameters and input variations using the variation block [54]. We select top k and last k sinks in nominal RC delay and measure the difference of delay variation between each pair of top k and last k sinks to estimate the worst case clock skew of the clock mesh under the limit of DCMatch. When measuring more than 9 output variables, the error is less than 0.42ps and the analysis takes more than 81s. Although the DCMatch is very fast than Monte-Carlo simulation, it still takes a few seconds for the smallest benchmark circuit (s9234). For the accuracy, we measure 30 outputs, which is maximum allowed number of output variable of a Hspice DCMatch analysis, for each DCMatch analysis for the following experiment.

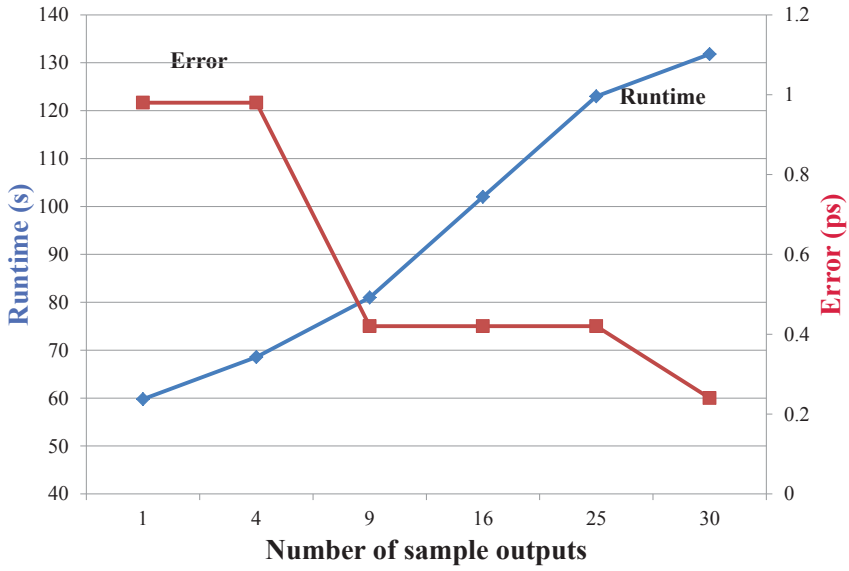


Figure 4.13: The changes of runtime (blue curve) and error (red curve) as the number of sample outputs varies from 1 to 30 for s9234 [2].

We analyze the trade-off between the worst case clock skew and the power reduction of CMESH-int (blue), MeshWorks (green), the network survivability based method in [3] (red), and DCMatch (purple) by varying the constraints of each methods in Fig. 4.14. The x-coordinate indicates the worst case clock skew measured by Hspice

Monte-Carlo simulation under the same variation environment used in Fig. 4.11. The y-coordinate indicates the power reduction which is also measured by Hspice simulation. For the result of DCMatch, we implement the clock mesh optimization using the result of the DCMatch analysis into our framework. We estimate the worst case clock skew of the clock mesh by DCMatch analysis for checking the feasibility of movements instead of clock skew and multiple path constraints which are used to check the feasibility of movements in the proposed framework CMESH-int. In each iteration, we sort the movement candidates in increasing order of the cost (Eq. 4.1) and check the worst case clock skew constraint of movement candidates one by one. If a movement satisfies the worst case lock skew constraint, then we apply the movement.

The result of DCMatch shows the maximum power reduction below the 36ps of worst case clock skew. The result of CMESH-int achieves similar power reduction to the result of the DCMatch while other works shows much lower power reduction. Meanwhile, the result of CMESH-int shows the best result above the 36ps of worst case clock skew. That is mainly caused by the timing error of the optimized result of DCMatch analysis is up to 10ps in comparison with result of Monte-Carlo simulation when we set the worst case clock skew constraint larger than 36ps. Through the experiments, we show that our approach produces very similar power reduction to result using DCMatch analysis while taking less time than DCMatch. Our approach takes 10.73s for s9234 maximally, but optimization using DCMatch takes about 8 hours maximally which is about 3000X larger than our approach.

4.5.4 Analysis of the Effect of Proposed Techniques

We present the trade-off between the worst case clock skew and the power reduction of the proposed framework without the fast estimation techniques (red), with the fast estimation techniques (blue), without the path length based stub rebinding (green) and without the multiple-path constraint (purple) by varying the variation tolerance con-

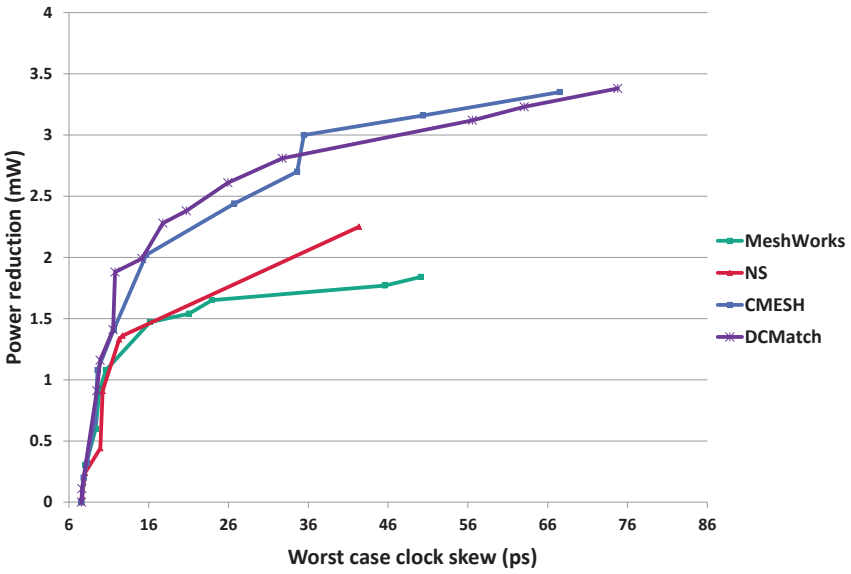


Figure 4.14: Trade-off between worst case clock skew and power reduction of CMESH-int, MeshWorks, network survivability based method [3], and DCMatch while varying the constraints of each methods for s9234 [2].

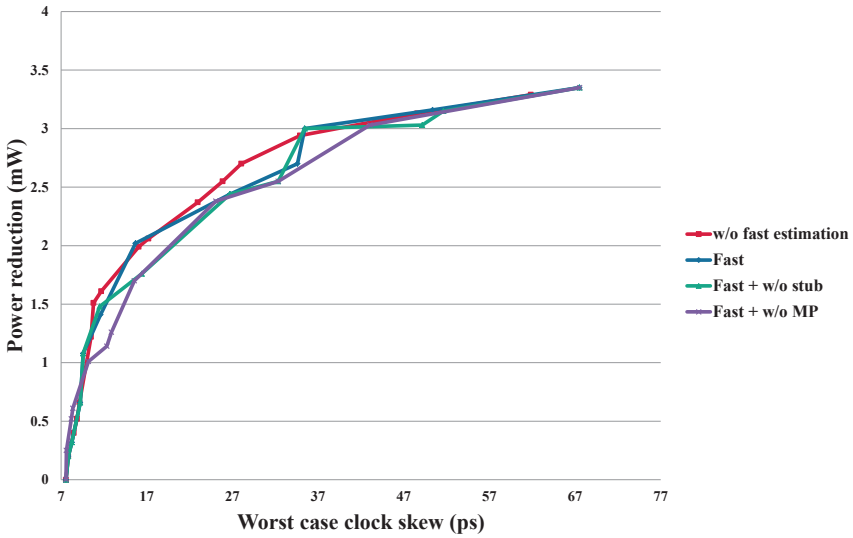
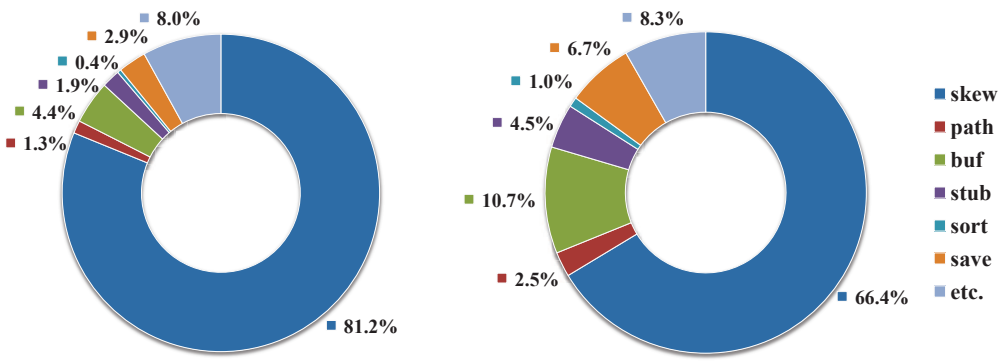


Figure 4.15: Trade-off between worst case clock skew and power reduction of the proposed framework without the fast estimation techniques (red), with the fast estimation techniques (blue), without the path length based stub rebinding (green) and without the multiple-path constraint (purple) while varying the variation tolerance constraints of the proposed methods for s9234 [2].

straints of the proposed method in Fig. 4.15. The x-coordinate indicates the worst case clock skew measured by Hspice Monte-Carlo simulation under the same variation environment used in Fig. 4.11. The y-coordinate indicates the power reduction which is also measured by Hspice simulation. The result of the four cases is similar when the constraints are highly tight and loose (i.e., under 10ps and over 35ps). The clock mesh optimization without the fast estimation techniques (red) shows largest power reduction while taking about 1.5X larger runtime than others. It reduces about 0.3mW, 0,4mW and 0.6mW more power maximally than the result applying the fast estimation techniques (blue), without the path length based stub rebinding (green) and the without multiple-path constraint (purple), respectively. The result without the multiple-path constraint (purple) shows lowest power reduction overall range of the worst case clock skew.

4.5.5 Run Time Analysis



(a) w/o fast estimation **(b) w/ fast estimation**

Figure 4.16: The breakdown graph of average runtime (a) without fast estimation and (b) with fast estimation for ISCAS89 benchmark circuits [2].

• *Runtime breakdown graph*: In Fig. 4.16, we show the breakdown graph of average runtime (a) without the fast estimation techniques and (b) with the fast estima-

tion techniques of ISCAS89 benchmark circuits [2]. We divide the runtime into seven categories which refer to runtime of clock skew estimation, calculating the degree of collective path redundancy, buffer resizing, path length calculation for stub rebinding, sorting, saving and loading of mesh state and etc. (e.g., preparing and verifying data structures). By applying the fast estimation techniques, the average runtime is reduced about 63% and the runtime ratio of clock skew estimation is also reduced from 81.2% to 66.4%.

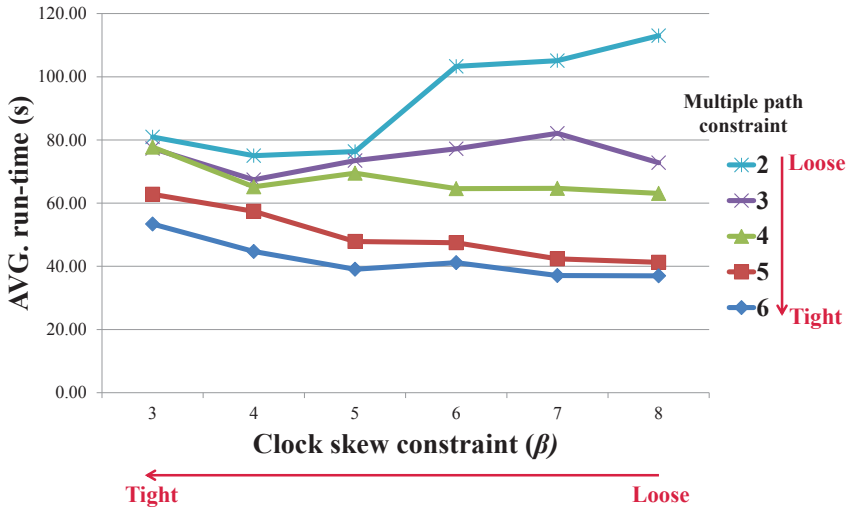


Figure 4.17: The run time analysis of CMESH-int by varying clock skew constraint and multiple path constraint. Y-coordinate indicates average run time of ISCAS89 benchmark circuits [2] under each constraint values.

• *Variation tolerance constraints*: Fig. 4.17 shows the average run time of six ISCAS89 benchmark circuits by varying clock skew and multiple path constraints. The x-coordinate indicates the clock skew constraint and the y-coordinate indicates the average run time of ISCAS89 benchmark circuits. The number on each curve represents the multiple path constraint used, which means *the degree of collective path redundancy* of every stub ($\rho(\cdot)$) should be larger than the number. As mentioned above, the

largest portion of the run time is taken in the clock skew calculation. Under a tight multiple path constraint (i.e., high number), many possible moves would not meet the multiple path constraint. Thus, the number of possible moves to check the clock skew constraint becomes small, resulting in a short run time. The vertical order of the five curves shows this trend. On the other side, if the clock skew of current clock mesh is much smaller than the clock skew constraint (i.e., under loose skew constraint), CMESH-int simply accepts the best move without checking the clock skew constraint, and checks the clock skew constraint once the application of the best move is done. (An undo operation will be performed only if it violates the clock skew constraint. However, as the clock skew is close to the clock skew constraint, our fast clock skew estimation technique is applied to every possible move before choosing the best move.) For mesh optimization with tight clock skew constraint, CMESH-int requires a relatively high run time (e.g., the leftmost part of the top two curves in Fig. 4.17) for loose multiple path constraint (e.g., 2 and 3) due to the existence of many possible moves.

• *Early termination of iterations*: In Fig. 4.18, we show the average run time (blue curve) and average total wirelength (red curve) of six designs [2] by varying the early termination threshold from 0.5 to 2.5 of CMESH-int. Other parameters are the same with that used in Table 4.2. The average run time increases by 45% while reducing the total wirelength by 4% by increasing the early termination threshold from 0.5 to 2.5. As a result, we can reduce the run time significantly at the expense of a small increase of total wirelength by applying the early termination scheme.

4.5.6 Accuracy and Run Time of Fast Clock Skew Estimation

In this subsection, we highlight and discuss, through a set of experiments, how well the proposed three acceleration techniques are effective in reducing the computation

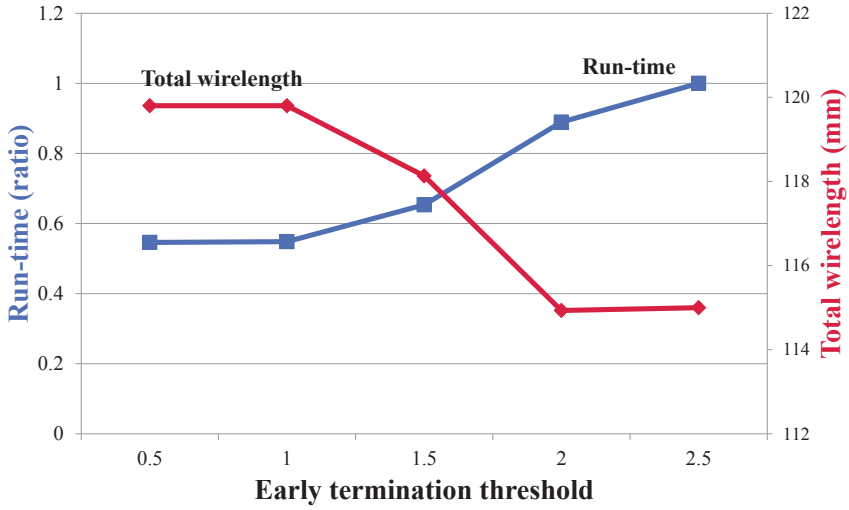


Figure 4.18: The changes of average run time (blue curve) and average total wirelength (red curve) as the early termination threshold varies from 0.5 to 2.5 of CMESH-int for ISCAS89 benchmark circuits [2].

time.

• *Our modified 1-π RC circuit model*: We analyze the accuracy of clock skew and the runtime of our modified RC circuit model comparing with Hspice transient analysis and dc analysis on ISCAS89 benchmark circuits in Table 4.7. The clock skew of our model is calculated using UMFPACK [51]. The clock skew analysis using our modified RC circuit model takes 4.97s and 1.24s less time than Hspice transient and dc analysis on average, respectively. The average clock skew error of our model is 0.31 ps comparing with Hspice transient analysis.

• *Reuse of matrix multiplication*: Fig. 4.19 shows comparisons of the computation times of the clock skew estimation by reusing (denoted by Reuse) and without (denoted by No-reuse) reusing the result of LU decompositions. The x-coordinate indicates the numbers of mesh segments that correspond to trials of incremental update in the *inner-loop* of CMESH-int for four designs [2]. The y-coordinate indicates the

Table 4.7: Clock skew and runtime of our model, Hspice transient analysis and hspice dc analysis on ISCAS89 benchmark circuits [2]

Circuit	Our model		Hspice transient		Hspice dc	
	skew (ps)	runtime (s)	skew (ps)	runtime (s)	skew (ps)	runtime(s)
s9234	1.11	0.01	1.26	2.25	1.11	0.8
s13207	2.09	0.02	2.46	3.55	2.09	0.91
s15850	2.02	0.02	2.29	3.29	2.02	0.87
s35932	1.93	0.05	2.16	6.98	1.93	1.83
s38417	3.81	0.05	4.36	6.96	3.81	1.56
s38584	1.77	0.04	2.05	6.99	1.77	1.62
Avg.	2.12	0.03	2.43 (+0.31)	5.0 (+4.97)	2.12 (+0.00)	1.27 (+1.24)

normalized run time corresponding to Reuse (blue) and No-reuse (red). In summary, Reuse spends only $20\% \sim 38\%$ of the run time of No-reuse and 27% of the run time of No-reuse on average. In addition, as the number of mesh segments for incremental updates increases, the run time saving tends to increase as well.

• *Sliding window*: Under the reuse of matrix multiplications and delay update with $\epsilon = 0.01ps$, we applied CMESH-int to designs [2] by varying the window size, and summarized the result in Fig. 4.20. Fig. 4.20(a) shows the average values of small circuits s13207 and s15850 while Fig. 4.20(b) shows the average values of large circuits s35932, s38417 and s38584. The x-coordinate indicates the number of windows that cover the clock mesh. The left and right y-coordinates indicate the normalized clock skew computation time and the clock skew difference in pico seconds with respect to that produced by using a single window scheme on the clock mesh. Clearly, the clock skew error increases as more and smaller sliding windows are used. It is expected that the run time decreases as the number of windows increases, but it is shown that the run time increases when the number of windows is over 36 of Fig. 4.20(a) and over 100 of

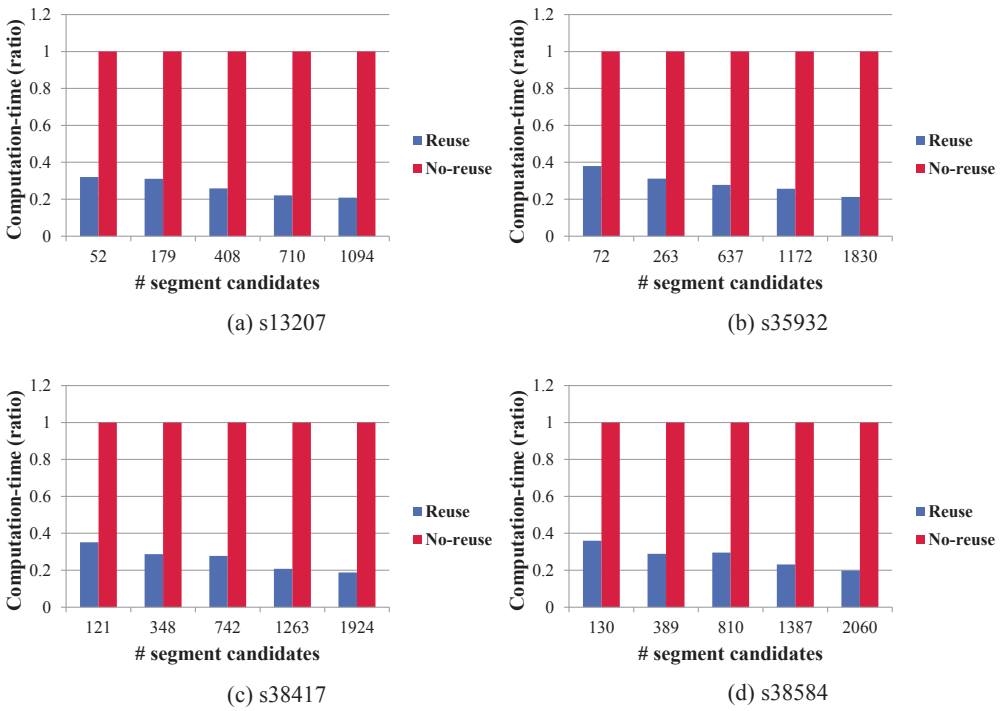
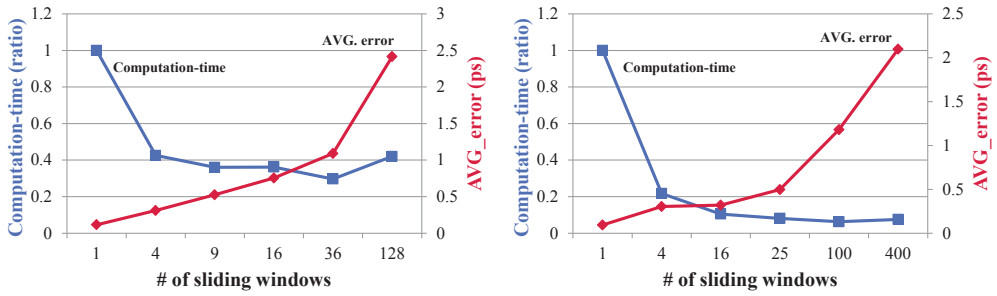


Figure 4.19: Run time comparison for the clock skew estimation by CMESH-int when the result of LU decompositions is reused (denoted by Reuse) and not reused (denoted by No-reuse) for four designs [2].

Fig. 4.20(b). This is because the total time increase in processing many window borders is more than the total time saving in processing the small windows. In addition, it shows that the sliding window scheme can achieve high run time reduction for large circuits. The run time ratio is 0.3∼0.43 for small circuits (Fig. 4.20(a)), and the run time ratio is 0.06∼0.21 for large circuits (Fig. 4.20(b)).



(a) Small circuits

(b) Large circuits

Figure 4.20: The changes of average run time (blue curve) and average clock skew error (red curve) as the size of sliding window varies from the largest to the smallest for designs in [2].

• *Delay updating caused by buffer resizing*: Under the reuse of matrix multiplications and a single window scheme, we applied CMESH-int to six designs [2] by varying the threshold parameter ϵ, which was used as a termination condition for the delay adjustment caused by buffer resizing. The results are depicted in Fig. 4.21 where the left and right y-coordinates indicate the normalized average clock skew computation time and the average clock skew difference in pico seconds with respect to that produced by the conventional RC delay calculation to the entire RC network for the delay adjustment. The two curves show a well trade-off between run time and accuracy of clock skew estimation.

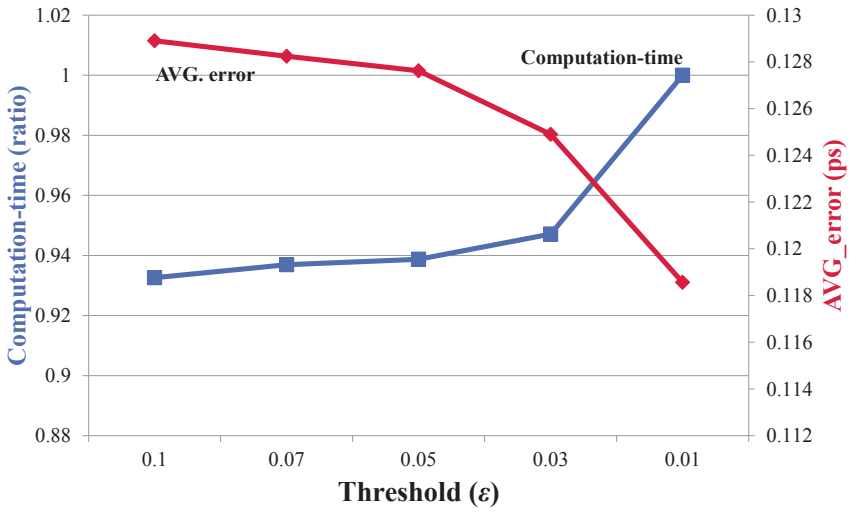


Figure 4.21: The changes of average run time (blue curve) and average clock skew error (red curve) as the threshold parameter ϵ varies from 0.1 to 0.01 for designs in [2].

4.5.7 Electromigration Analysis

Although the clock mesh optimization reduces the unnecessary power consumption of the clock mesh, the clock mesh optimization can potentially enlarge current density of mesh segments which are near to deallocated mesh segments. The high current density on a metal wire causes displacement of metal atoms which is called electromigration (EM). Electromigration can increase the resistance of metal wires and eventually incur failure of the wires. The empirical equation of mean time to failure (MTTF) considering the effect of electromigration of a wire was suggested in [55].

$$MTTF = \frac{A}{J_{avg}^{n}} \cdot exp(\frac{E_a}{k \cdot T}) \tag{4.7}$$

where A is the empirical constant based on cross-sectional of the wire, J_{avg} is the average current density, E_a is the activation energy, k is the *Boltzmann's constant*, T is the temperature in *Kelvin* and n is the empirical constant set to 2 in [55].

Among the parameters in *Eq. 4.7*, only the average current density (J_{avg}) can be

efficiently controlled by wire sizing [4, 56]. The iterative wire sizing method was suggested in [4] to fix the EM violations after the clock mesh optimization. In which, they increase the width of mesh segments in linear proportion to the magnitude of EM violation, and then perform the mesh buffer resizing. They run the above process iteratively until fixing all EM violations. In this work, we present the power increment by performing EM fixing method in [4].

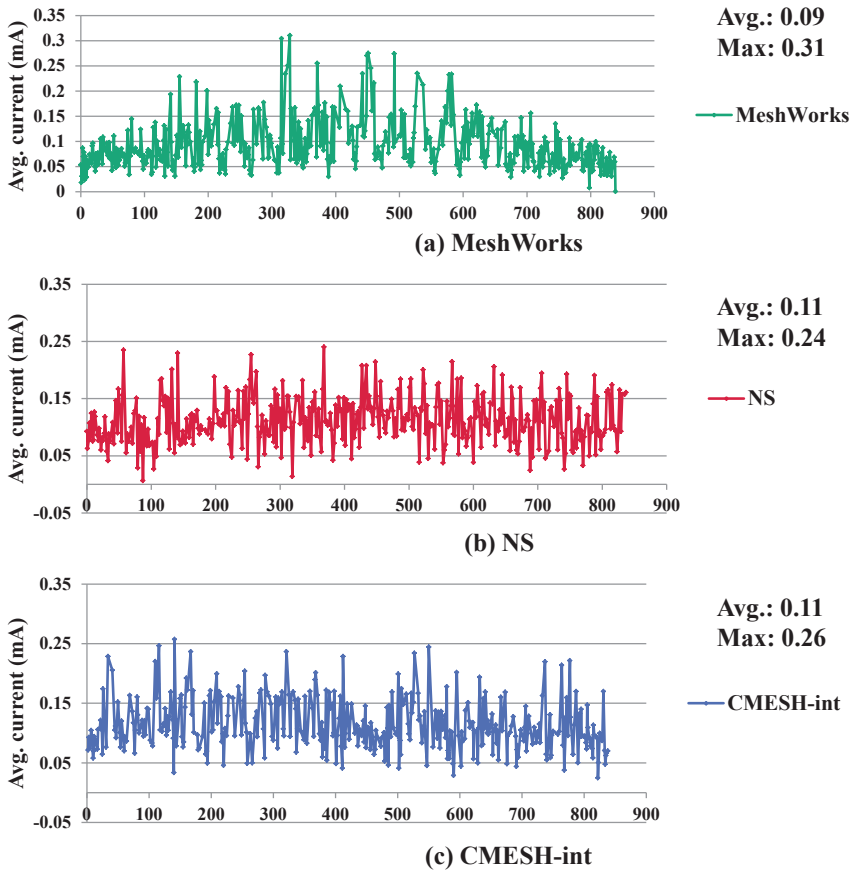


Figure 4.22: The average current of mesh segments of optimized results produced by (a) MeshWorks, (b) NS and (c) CMESH-int for s38417 in [2].

The average current of mesh segments of the optimized clock mesh produced by (a)

MeshWorks (green), (b) NS (red) and (c) CMESH-int (blue) is presented in Fig. 4.22. All parameters are identical to that in Fig. 4.11. The average current of mesh segments is measured by Hspice simulation. The maximum value of average current of MeshWorks is 0.31 mA which is 0.07 mA and 0.05 mA larger than NS and CMESH-int, respectively. Since the NS and ours provide certain level of redundancy on each clock sink by network survivability based constraint and multiple-path constraint, the two algorithms shows the lower average current than MeshWorks.

We perform the EM fixing algorithm in [4] on results produced by MeshWorks (green), network survivability based method in [3] (red) and CMESH-int (blue) by varying the EM constraint from 3 to 10 in Fig. 4.23. We measure the average current of each mesh segments and power consumption of the clock mesh by Hspice simulation. The x-coordinate indicates the EM constraint and y-coordinate indicates the average power increment after applying EM fixing method. The dashed lines in Fig. 4.23 mean that some benchmarks are not fixed by EM fixing method. The power increment of our approach is smallest until 4 $mA/\mu m^2$ of EM constraint with less than 1% of power overhead. 4 $mA/\mu m^2$ is tight constraint than the recommended values in [4, 57, 58] which are 7.2, 20 and 30 $mA/\mu m^2$. Therefore, the results of our approach can be fixed with minimum power increment in reasonable EM constraint range.

4.5.8 Run-time Analysis in Multi-thread Computing Environment

If the computing environment with parallel processing is available, the task of sliding window scheme performed in CMESH-int could be partitioned into several independent sub-tasks to be executed in parallel since the perturbation candidates in each window are independent in the calculation of both the clock skew and the number of multiple paths. We implement multi-threading using OpenMP on our clock mesh optimization. We perform the proposed approach on ISCAS89 benchmark circuits by varying the number of thread in Fig. 4.24. The x-coordinate indicates the number of thread and

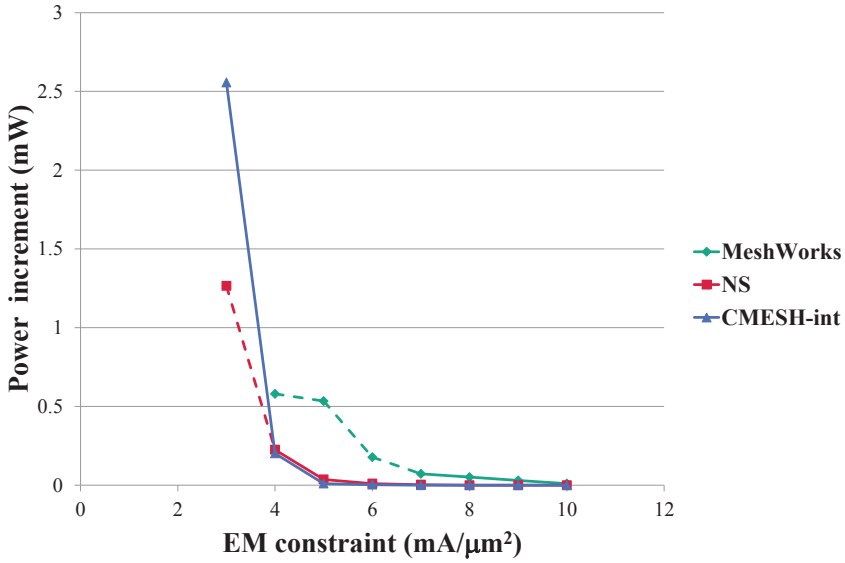


Figure 4.23: The changes of average power increment by applying EM fixing on results of MeshWorks, NS(Network Survivability based method) and CMESH-int as average current density constraint varies from 3 to 10 for designs in [2].

y-coordinate indicates the runtime reduction of each benchmark circuits. We achieve 28.9% and 45.9% of the runtime reduction in average when using 2 and 4 of threads, respectively. More than 4 threads doesn't improve the runtime reduction, since in our approach after applying one movement, we re-check the clock skew and multiple path constraint for nearby sliding windows which is 4 sliding windows at most.

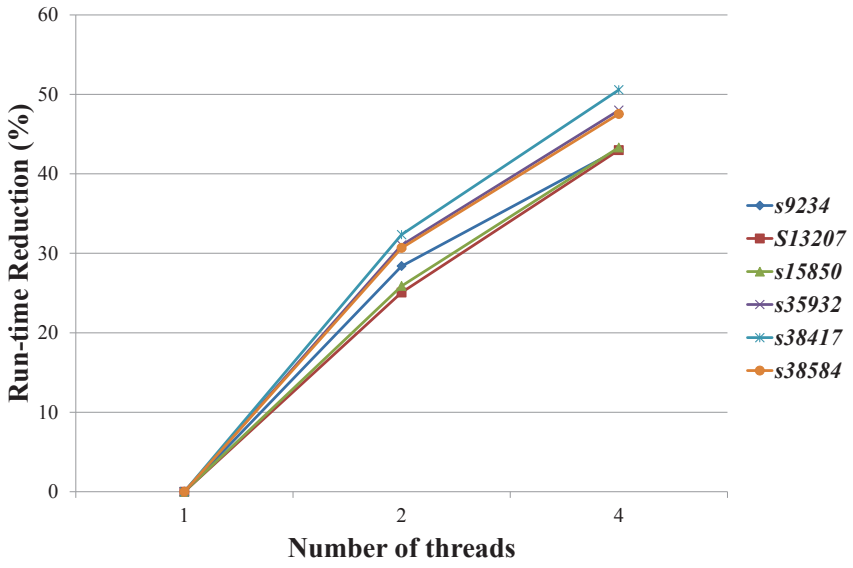


Figure 4.24: The run-time reduction in multi-threads computing environment as number of threads varies from 1 to 4 for designs in [2].

4.6 Summary

In this work, we addressed a comprehensive framework to solve the problem of clock mesh optimization which includes (1) mesh segment allocation, (2) stub wire binding, and (3) mesh buffer sizing. Unlike the conventional works which have performed the three tasks *sequentially*, we proposed an integrated clock mesh optimization algorithm by combining the three tasks into an iterative framework of incremental updates to find

a globally optimal resource allocation and binding of the clock mesh while considering the clock skew tolerance constraints. Additionally we fully exploited a set of fast clock skew estimation techniques to relieve the runtime complexity of the proposed framework.

Chapter 5

Conclusion

This dissertation proposed a comprehensive framework to solve the problem of clock mesh resource allocation and binding. Unlike the conventional approaches which have performed the tightly inter-related resource allocation and binding tasks namely (1) mesh segment allocation, (2) stub wire binding, and (3) mesh buffer sizing *sequentially* due to the inherently high computational complexity of the problem, this work addressed the problem in an integrated fashion by combining the three tasks into an iterative framework of incremental updates and *solving them simultaneously* to find a globally optimal allocation of mesh resources while taking into account the clock skew tolerance constraints. The core parts of this work were a precise analysis on the relation among the resource optimization tasks and an establishment of mechanism for effective and efficient integration of the tasks. In particular, to cope with the run time problem, we proposed a set of acceleration techniques to fit into our context of fast clock skew estimation in mesh resource optimization as well as invented early decision policies. Through experiments with ISCAS89 and ISPD2010 benchmark circuits, our proposed clock mesh resource framework was able to produce clock meshes with 13.2% less wirelength, 10.9% smaller buffer area, and 11.0% less power consumption

with 12.7% less short circuit power over the best known existing results. Furthermore, our framework was able to reduce the worst clock skew by $25 \sim 27\%$.

Bibliography

[1] M. Edahiro, "A clustering-based optimization algorithm in zero-skew routings," in *Proceedings of the 30th ACM/IEEE Design Automation Conference*, Jun. 1993, pp. 612–616.

[2] "Placement of iscas89 benchmark circuits," 1989, available:http://www.ece.wisc.edu/vlsi/tools/iscas-placement/index.html.

[3] G. Venkataraman, J. H. Z. Feng, and P. Li, "Combinatorial algorithms for fast clock mesh optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 1, pp. 131–141, Jan. 2010.

[4] A. Rajaram and D. Z. Pan, "Meshworks: a comprehensive framework for optimized clock mesh network synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 1945–1958, Dec. 2010.

[5] H. Chen and et al., "A sliding window scheme for accurate clock mesh analysis," in *Proceedings of the ACM/IEEE International Conference on Computer-Aided Design*, Nov. 2005, pp. 939–946.

[6] ISPD, "ISPD 2010 clock network synthesis contest," 2010, available:http://archive.sigda.org/ispd/contests/10/ispd10cns.html.

[7] N. Z. Haron and S. Hamdioui, “Why is cmos scaling coming to and end?” in *Proceedings of the 3rd International Design and Test Workshop (IDT)*, Dec. 2008, pp. 98–103.

[8] V. Mehrotra, “Modeling the effect of systematic process variation on circuit performance,” *Ph.D. Thesis, MIT*, May 2001.

[9] X. Lu, Z. Li, W. Qui, D. M. H. Walker, and W. Shi, “Longest-path selection for delay test under process variations,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 12, pp. 1924–1929, Dec. 2005.

[10] M. Alioto, G. Palumbo, and M. Pennisi, “Understanding the effect of process variations on the delay of static and domino logic,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 5, pp. 697–710, May 2010.

[11] J. R. Lorch and A. J. Smith, “Reducing processor power consumption by improving processor time management in a single-user operating system,” in *Proceedings of the Second ACM International Conference on Mobile Computing and Networking*, Nov. 1996, pp. 143–154.

[12] D. Panigrahi, C. Chiasserini, S. Dey, and R. Rao, “Battery life estimation of mobile embedded system,” in *Proceedings of 4th International Conference on VLSI Design*, Jan. 2001, pp. 57–63.

[13] R. Mahajan, C.-P. Chiu, and G. Chrysler, “Cooling a microprocessor chip,” *Proceedings of the IEEE*, vol. 94, no. 8, pp. 1476–1486, Aug. 2006.

[14] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin, “Dynamic thermal management through task scheduling,” in *Proceedings of IEEE International Symposium on Performance Analysis of System and Software*, Apr. 2008, pp. 191–201.

[15] P. E. Gronowski, W. J. Bowhill, R. P. Preston, M. K. Gowan, and R. L. Allmon, "High-performance microprocessor," vol. 33, no. 5, pp. 676–686, May 1998.

[16] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez, "Reducing power in high-performance microprocessors," in *Proceedings of the ACM/IEEE Design Automation Conference*, Jun. 1998, pp. 732–737.

[17] A. B. Kahng and C.-W. A. Tsao, "Practical bounded-skew clock routing," *Journal of VLSI Signal Processing for Signal, Image and Video Technology*, vol. 16, no. 2-3, pp. 199–215, Jun. 1997.

[18] J. G. Xi and W. W.-M. Dai, "Useful-skew clock routing with gate sizing for low power design," *Journal of VLSI Signal Processing for Signal, Image and Video Technology*, vol. 16, no. 2-3, pp. 163–179, Jun. 1997.

[19] M. R. Guthaus, X. Hu, G. Wilke, G. Flach, and R. Reis, "High-performance clock mesh optimization," *ACM Transactions on Design Automation of Electronic Systems*, vol. 17, no. 3, pp. 33:1–33:17, Jun. 2012.

[20] J. G. Xi and W. W.-M. Dai, "Buffer insertion and sizing under process variation for low power clock distribution," in *Proceedings of the ACM/IEEE Design Automation Conference*, Jun. 1995, pp. 491–496.

[21] A. Vittal and M. M.-Sadowska, "Power optimal buffered clock tree design," in *Proceedings of the ACM/IEEE Design Automation Conference*, Jun. 1995, pp. 497–502.

[22] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on rtl clock-gating," in *Proceedings of the ACM/IEEE Design Automation Conference*, Jun. 2003, pp. 622–627.

[23] R. Bhutada and Y. Manoli, “Complex clock gating with integrated clock gating logic cell,” in *Proceedings of International Design and Technology of Integrated Systems in Nanoscale Era*, Sep. 2007, pp. 164–169.

[24] X. Hu, W. Condley, and M. R. Guthaus, “Library-aware resonant clock synthesis (larcs),” in *Proceedings of the ACM/IEEE Design Automation Conference*, Jun. 2012, pp. 145–150.

[25] V. S. Sathe, S. Arekapudi, A. Ishii, C. Ouyang, M. C. Papaefthymiou, and S. Naffziger, “Resonant-clock design for a power-efficient high-volume x86-64 microprocessor,” vol. 48, no. 1, pp. 140–149, Jan. 2013.

[26] P. Restle, D. Shan, D. Hogenmiller, Y. Kim, A. Drake, J. Hibbeler, T. Bucelot, G. Still, K. Jenkins, and J. Friedrich, “Wide-frequency-range resonant clock with on-the-fly mode changing for the power^{tm} microprocessor,” in *Proceedings of IEEE International Solid-State Circuits Conference Digest of Technical Papers*, Feb. 2014, pp. 100–101.

[27] M. A. B. Jackson, A. Srinnivasan, and E. S. Kuh, “Clock routing for high-performance ics,” in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, Jun. 1990, pp. 573–579.

[28] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng, “Zero-skew clock routing with minimum wirelength,” *IEEE Transactions on Circuits and Systems*, vol. 39, no. 11, pp. 799–814, Nov. 1992.

[29] R.-S. Tsay, “An exact zero-skew clock routing algorithm,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 2, pp. 242–249, Feb. 1993.

[30] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao, "Bounded-skew clock and steiner routing," *ACM Transactions on Design Automation of Electronic Systems*, vol. 3, no. 3, pp. 341–388, Jul. 1998.

[31] C.-W. A. Tsao and C.-K. Koh, "Ust/dme: A clock tree router for general skew constraints," *ACM Transactions on Design Automation of Electronic Systems*, vol. 7, no. 3, pp. 359–379, Jul. 2002.

[32] G. E. Tellez and M. Sarrafzadeh, "Minimal buffer insertion in clock trees with skew and slew rate constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 4, pp. 333–342, Apr. 1997.

[33] Y. Liu, S. R. Nassif, L. T. Pileggi, and A. J. Strojwas, "Impact of interconnect variations on the clock skew of a gigahertz microprocessor," in *Proceedings of the ACM/IEEE Design Automation Conference*, Jun. 2000, pp. 168–171.

[34] S. Zanella, A. Nardi, A. Neviani, M. Quarantelli, S. Saxena, and C. Guardiani, "Analysis of the impact of process variations on clock skew," *IEEE Transactions on Semiconductor Manufacturing*, vol. 13, no. 4, pp. 401–407, Nov. 2000.

[35] E. Malavasi, S. Zanella, and M. Cao, "Impact analysis of process variability on clock skew," in *Proceedings of the 3th International Symposium on Quality Electronic Design*, Mar. 2002, pp. 129–132.

[36] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical clock skew analysis considering intra-die process variations," in *Proceedings of the ACM/IEEE International Conference on Computer-Aided Design*, Nov. 2003, pp. 914–921.

[37] S. Bhunia and S. Mukhopadhyay, *Low-Power Variation-Tolerant Design in Nanometer Silicon*. Springer, 2011.

[38] V. Wason, R. Murgai, and W. W. Walker, "An efficient uncertainty- and skew-aware methodology for clock tree synthesis and analysis," in *Proceedings of the 20th International Conference on VLSI Design*, Jan. 2007, pp. 271–277.

[39] U. Padmanabhan, J. M. Wang, and J. Hu, "Robust clock tree routing in the presence of process variations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 8, pp. 1385–1397, Aug. 2008.

[40] A. Rajaram, J. Hu, and R. Mahapatra, "Reducing clock skew variability via crosslinks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 1176–1182, Jun. 2006.

[41] J.-S. Yang, A. Rajaram, N. Shi, J. Chen, and D. Z. Pan, "Sensitivity based link insertion for variation tolerant clock network synthesis," in *Proceedings of the 8th International Symposium on Quality Electronic Design*, Mar. 2007, pp. 398–403.

[42] W.-C. D. Lam, J. Jain, C.-K. Koh, V. Balakrishnan, and Y. Chen, "Statistical based link insertion for robust clock network design," in *Proceedings of the ACM/IEEE International Conference on Computer-Aided Design*, Nov. 2005, pp. 588–591.

[43] R. J. Restle and et al., "A clock distribution network for microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 5, pp. 792–799, May 2001.

[44] M. Cho, D. Z. Pan, and R. Puri, "Novel binary linear programming for high performance clock mesh synthesis," in *Proceedings of the ACM/IEEE International Conference on Computer-Aided Design*, Nov. 2010, pp. 438–443.

[45] P. Chakrabarti, V. Bhatt, D. Hill, and A. Cao, "Clock mesh framework," in *Proceedings of the IEEE International Symposium on Quality Electronic Design*, Mar. 2012, pp. 381–386.

[46] J. Lu, X. Mao, and B. Taskin, "Integrated clock mesh synthesis with incremental register placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 2, pp. 217–227, Feb. 2012.

[47] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 49, no. 1, pp. 291–307, 1970.

[48] M. Celik, L. Pileggi, and A. Odabasioglu, *IC Interconnect Analysis*. Springer, 2002.

[49] R. A. Rohrer, "Circuit partitioning simplified," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 1, pp. 2–5, Jan. 1988.

[50] Y. Zhong and M. D. F. Wong, "Fast placement optimization of power supply pads," in *Proceedings of the 12th Asia and South Pacific Design Automation Conference*, Jan. 2007, pp. 763–767.

[51] T. A. Davis, "A column pre-ordering strategy for the unsymmetric-pattern multifrontal method," *ACM Transactions on Mathematical Software*, vol. 30, no. 2, pp. 165–195, Jun. 2004.

[52] "Predictive technology model," 2011, available:http://ptm.asu.edu/.

[53] H. Chang and S. S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 9, pp. 1467–1482, Sep. 2005.

[54] Synopsys, *Hspice(R) User Guide: Simulation and Analysis. Version E-2010.12*, Dec. 2010.

[55] J. R. Black, “Electromigration – a brief survey and some recent results,” *IEEE Trans. on Electron Devices*, vol. ED-16, no. 4, pp. 338–347, Apr. 1969.

[56] M. P. Desai, R. Cvijetic, and J. Jensen, “Sizing of clock distribution networks for high performance cpu chips,” in *Proceedings of the 33th ACM/IEEE Design Automation Conference*, 1996, pp. 389–394.

[57] S. M. Alam, F. L. Wei, C. L. Gan, C. V. Thompson, and D. E. Troxel, “Electromigration reliability comparison of cu and al interconnects,” in *Proceedings of the 6th International Symposium on Quality Electronic Design (ISQED)*, Mar. 2005, pp. 303–308.

[58] D. Wang, A. Y. Zhao, L. Yu, J. Wu, V. Chang, and W.-T. K. Chien, “Early failure model analysis and improvement of the upstream electromigration in 45nm cu low-k interconnects,” in *2013 IEEE International Reliability Physics Symposium (IRPS)*, Apr. 2013, pp. EM.1.1–EM.1.3.

국문 초록

클락 네트워크는 동기화된 디지털 회로의 모든 저장셀들에 클락 신호를 전달한다. 반도체공정이 미세화됨에 따라 공정, 전압, 온도 (PVT) 변이가 증가하고, 이로 인한 클락 스큐 변이 증가는 회로의 성능을 저하시키고 회로 오동작의 주요 원인이 되고 있다. 최근 클락 스큐 변이를 감소시키기 위한 방법 중에 하나로 클락 메쉬에 대한 연구가 활발히 진행되고 있다. 클락 메쉬는 변이 내성이 높지만 많은 양의 추가 와이어와 버퍼 사용하고 이는 클락 네트워크의 파워 소모 증가로 이어진다. 따라서 클락 메쉬의 변이 내성을 유지하면서 자원 사용을 최적화하는 클락 메쉬 합성 기법에 대한 연구가 필요하다. 클락 메쉬 합성 결과에 중요한 영향을 주는 3가지 테스크는 다음과 같다. *(1)메쉬 세크먼트 할당, (2) 메쉬 버퍼 할당과 크기 결정, (3) 메쉬 세크먼트에 클락 싱크 바인딩.* 기존의 클락 메쉬 최적화 연구에서는 수행시간을 최소화하기 위하여 이 3가지 테스크를 순차적으로 수행하였다. 하지만 이 3가지 테스크는 서로 조그만 변화가 다른 테스크의 결과에 큰 영향을 줄 수 있기 때문에 한 번에 모두 고려하는 방법이 필요하다. 본 논문에서는 위의 3가지 테스크를 결합하여 반복적으로 수행하여 클락 스큐의 변이 내성 제한을 만족 시키면서 클락 메쉬의 전체 자원 사용을 최적화하는 방법을 제시한다. 특히 최적화 테스크들 사이의 관계를 정밀히 분석하고 효과적으로 결합하여 수행하기 위한 방법을 제안한다. 추가적으로, 수행 시간의 커지는 문제를 해결하기 위하여 클락 메쉬의 자원 최적화 과정에 클락 스큐 추정을 위한 속도 향상 기법과 초기 결정 정책을 제안하고 적용한다. 요약하면, 본 논문에서는 테스크 결합과 수행시간 최소화 문제를 고려한 효과적인 메쉬 기반의 클락 네트워크 합성 기법을 제시하였다.

주요어: VLSI&CAD, 클락 메쉬 합성, 변이에 강한 설계, 자원 할당과 바인딩

학번: 2010-30206